Software Non-functional Assessment Process (SNAP)

Assessment Practices Manual

Release 2.3

This page intentionally left blank

International Function Point Users Group (IFPUG)

SNAP Assessment Practices Manual

Release 2.3

Chairperson, Non-Functional Sizing Standards Committee (NFSSC)
Talmon Ben-Cnaan
NFSSC@ifpug.org

SNAP Assessment Practices Manual by International Function Point Users Group (IFPUG) is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License.
Based on a work at www.ifpug.org.
Permissions beyond the scope of this license may be available at www.ifpug.org.

ISBN 978-0-9903007-3-1

This page intentionally left blank

Release 2.3, May 2015

This release replaces Release 2.2, which is now obsolete.
Changes are made periodically to the information within.

Documentation Team

NFSS Committee

Dácil Castelo, IFPUG Director, Board Liaison

Talmon Ben-Cnaan, Amdocs (Chairperson)
Kathy Lamoureaux, HP Enterprise Services (Vice Chairperson)

Abinash Sahoo, Amdocs; Roopali Thapar, IBM; Charley Tichenor, DCA; Jalaja Venkat, iGATE; Luigi Buglione, Engineering Ingegneria Informatica SpA; Mauricio Aguiar, TI Metricas; M. George Mitwasi, Optum, Pablo Garcia Soneira, Sopra; Julian Gomez Bejarano Francisco, Sopra; Saurabh Saxena, Amdocs; Robert Bell, HP.

SNAP Assessment Practices Manual Team 2011-2012

Christine Green, IFPUG Director, Board Liaison

Steve Chizar, Retired, Naval Systems Supply Business Solutions, Chair

Abinash Sahoo, Amdocs, Vice Chairperson

Charlene Zhao, BMO; Charley Tichenor, DSCA; Jalaja Venkat, iGATE; Joanna Soles, CSC; Kathy Lamoureaux, HP Enterprise Services; Peter R Hill, ISBSG; Roopali Thapar, IBM; Talmon Ben-Cnaan, Amdocs

IT Performance Committee

Dan Bradley, Chair

Christine Green, HP Enterprise Services, SNAP Project Manager; Wendy Bloomfield, Great-West Life Assurance Co.;

Talmon Ben-Cnaan, Amdocs; Joanna Soles, CSC; Loredana Fralliccardi, CSC, IFPUG Director, SNAP Sponsor (2008-2010); Janet Russac, Software Measurement Expertise, Inc., IFPUG Director, SNAP Sponsor (2010-2011)

SNAP Assessment Practices Manual Team 2009-2011

Patty Boyce, NASCO; Sergio Brigido, HP Enterprise Services; Steve Chizar, Retired, Naval Systems Supply Business Solutions; Sheila Dennis, The David Consulting Group; David Garmus, The David Consulting Group; Asha Goyal, Nucleus Software Exports Ltd; Claudia Hazan, Employee of a Brazilian Gov IT Organization; Peter Hill, ISBSG; Lori Holmes, Q/P Management Group, Inc.; Ashwin Krishnamurthy, IBM India Pvt Ltd; Monica Lelli, IT Consultant; Luca Santillo, Agile Metrics; Jalaja Venkat, iGATE; Charles Wesolowski, QinetiQ North America Systems Engineering Group

SNAP Framework Team 2008

Albert Meyn, W&W Informatik GmbH; Ana Miccolis Petrobras; Ashwin Krishnamurthy, IBM; Carolyn Melvin, Northrop Grumman; Charles Wesolowski, Lockheed Martin Space Systems Company; David Seaver, PRICE Systems LLC; David Garmus, The David Consulting Group; Graeme Prescott, Australian Government; Jalaja iGATE Lori Holmes, Q/P Management Group, Inc.; Marco Motta, Vodafone; Mick Burn-Murdoch, Lalcrest Ltd; Mike Pearl, EDS; Muhammet Oeztuerk, W&W Informatik GmbH; Peter Grüner, G4Q IT Services; Pranabendu Bhattacharyya, Tata Consultancy Services

Special Thanks

A special thanks is offered to the team members of the two projects that have worked on defining the SNAP Framework as well as defining and creating the Assessment Practices Manual (APM), including:

- the two teams,
- all of the individuals and organizations participating in review, beta test, and feedback, and

- Bonnie Brown, and Luc Vangrunderbeeck, for their support with the APM layout, the beta test tool, and the analysis of beta test data

SNAP Reviewers – Thank You!

Ajith PN, Alain Gontard, Albert Meyn, Alcione Jandir Candéas Ramos, Amith Kulkarni, Anil Joshi, Atluri Kumar, Bonnie Brown, Candéas Ramos, Checng Cheng Zhao, Connie Smith, Dan Schultz, Dimple Gupta, Elisa Gallo, Emilia Leboreiro, Ganeshan Guruswamy, Giselle Fróes, Gurpreetkaur Marve, Guruprasath Sethumadhavan, Jay Fischer, Jaya Venkat, Jeff Lindskoog, Jim Watson, Joel Andre, Julian Gómez Bejarano, Jyoti Namjoshi, Kareem Quereshi, Kathy Lamoureaux, Krupanandra Babu, Krzysztof Kłosowski, Lionel Perrot, Luigi Buglione, Luis Andasola, Luis Carlos Grande, Luís Geminianno, Luz Ibáñez, Manoj Ramchandra Sable, Marcelo Leme, Marcin Soboń, Mario Vacalebri, Michael Fuller, Muhammet Oezteurk, Namit Mishra, Neetika Jain, Nicolas Chantereau, Nitin S Dharrao, Pablo Soneira, Padmanabhan Kalyanasundaram, Parag Saha, Peter Grüner, Peter Thomas, Philippe Perrin, Piotr Popowski, Pranabendu Bhattacharyya, Radhika Srinivas, Rafał Wasilewski, Rajib Banerjee, Ranjana Jain, Ravindra Moghe, Renata Faray, Rocío Juárez, Sachin Thakur, Savitha G, Sherry Ferrel, Shreyas Bhargave, Siddharth Misra, Sridhar Maheswaram, Srividhya Ganesan, Stanley Richardson, Susan Walls, Sushmitha Anantha ,Swapnil Dambe, Tomasz Gąsiorowski, Tomasz Marchel, Usha Yegnaseshan, Valkal Bairagi, Viral Shah, Wojciech Dratnal

Special thanks to Joe Schofield for style review

SNAP Beta test and review companies – Thank You!

Accenture, ALLIGRA (groupe Astek), AMDOCS, Banco Bradesco S/A, Banco do Brasil , Banco de Crédito del Perú, BRQ IT Solutions, Business Integration Partners S.p.a., Centurylink , Cognizant Technology Solutions, CSI Piemonte, Engineering.IT SpA , FÓTON INFORMÁTICA S.A, GE Energy, GDF Suez , HP Enterprise Services, IBM , Kubota Systems Inc., Level 4 Ventures, Inc., MindTree, Mphasis an HP Company, Neoris, Orange Polska, SEMANTYS (Groupe Astek), SIDI srl, Sopra Group, Tata Consultancy Services Ltd, TELUS, TI METRICAS SERVICOS LTDA, T-Systems, Austria,UST global

For information about additional copies of this manual, contact
IFPUG
191 Clarksville Road
Princeton Junction, NJ 08550
U.S.A.
(609) 799-4900
E-mail: **ifpug@ifpug.org**
Web: http://www.ifpug.org

Table of Contents

	Preface		v
	Introduction to the Assessment Practices Manual		ix
Part 1	**The SNAP Method**		
	Chapter 1	Introduction to SNAP	1-1
	Chapter 2	SNAP Overview	2-1
	Chapter 3	Assessment Preparation	3-1
	Chapter 4	Determine Purpose, Scope, Boundary and Partition	4-1
	Chapter 5	Categories & Sub-categories	5-1
	Chapter 6	Calculate Non-functional Size (SNAP Points)	6-1
Part 2	**Examples**		
	Chapter 1	Examples	1-1
Part 3	**Appendices**		
	Appendix A	Glossary	A-1
	Appendix B	IFPUG APM Link to IFPUG CPM	B-1
	Appendix C	Index	C-1
	Appendix D	APM Changes after Release 1.0	D-1

This page intentionally left blank

 (cc) BY-NC-SA

Preface

Introduction

Having both functional size and non-functional size provides a more complete picture of software product development. The functional size is quantifiable and represents a good measure of the functional project/application size. Providing a quantifiable measure for the Non-Functional Requirements (NFR) allows organizations to build historical data repositories that can be referenced to assist in decision making for the technical and/or quality aspects of applications.

A non-functional assessment will assist Information Technology (IT) organizations in multiple ways. It will provide insight into projects and applications to assist in estimating and in the analysis of quality and productivity. Used in conjunction with function point analysis, the non-functional assessment provides information that can identify items impacting quality and productivity in a positive or negative way. Having this information enables software professionals to:

- better plan, schedule and estimate projects,
- identify areas of process improvement,
- assist in determining future technical strategies,
- quantify the impacts of the current technical strategies, and
- provide specific data when communicating non-functional issues to various audiences.

(cc) BY-NC-SA

SNAP History & Background

The IT Performance Committee (ITPC), changed to be the IT Measurement Analysis Committee (ITMAC), received approval from the IFPUG Board to proceed with the Technical Size Framework (TSF) Project at the 2007 ISMA conference. The goal of the project was to define a framework that would size the technical aspects of software development. The current issue with functional size is that it has not been suitable for sizing the technical requirements associated with a software development project. The focus of the project was to develop a separate technical size framework from the function point (FP) methodology as defined by IFPUG. Making this separate from the functional size measure would ensure that historical function point data could continue to be used. The defined framework had to be agreed to and supported by the IFPUG Board and the IFPUG members. The final product would define guidelines and rules for sizing the non-functional aspects of software development and would include non-functional aspects.

Release Draft

The work of the Software Non-functional Assessment Process (SNAP) project team resulted in a first draft version released in October 2009 for the purpose of getting it reviewed by a team of reviewers as well as the IFPUG Board, The IFPUG New Environment Committee and the IFPUG Counting Practice Committee. This was reflected in Release 0.1 (October 2009) of the International Function Point Users Group (IFPUG) SNAP Assessment Practices Manual (APM).

Release Beta

The work of the SNAP project team, along with the feedback from the reviewers resulted in a first Beta version released in November 2010 for the purpose of getting it reviewed by the Software Industry in general. This content was reflected in Release 1.0 BETA (November 2010) SNAP APM. The feedback from the first review resulted in changes to small parts of the APM.

A beta test was performed in January 2011 on the second beta version to confirm the process as well as to provide data for consolidation of the model. Small changes to the APM were made as well as confirmation of the SNAP Sizing Units used for most sub-categories. A few changes were made to the sub-categories, including adding one new sub-category.

An additional beta test was performed in May 2011 to get the final data for consolidation of the calculation of the model to a consistent size measure.

The beta test was performed by multiple companies from around the globe.

Release 1.0

The initial public release of the SNAP APM (September 2011) defined the concepts, processes, and rules for assessing non-functional software requirements. Release 1.0 of the APM resulted from years of work by the SNAP team and its expert contributors. This release included additions and corrections collected during the beta test period and additional input from the team.

Post release 1.0

Changes in the APM after release 1 are detailed in Appendix B.

Future Releases

This document is meant to be a living one. We must recognize how to perform assessments in new environments as they are introduced. We need to be able to do this in the context of maintaining the validity of the assessments we have already made. Sustaining this consistency may not be simple, yet it is essential if we are to be able to measure the progress we are making in delivering value to the users and to the organizations they represent.

The Non-Functional Standards Sizing Committee wishes to thank all those who have helped us in our research and in the production of this manual.

Talmon Ben-Cnaan

Chairperson, NFSSC.

 (cc) BY-NC-SA

This page intentionally left blank

Introduction to the Assessment Practices Manual

Introduction

This introduction defines the objectives of the manual and the revision process. It also describes publications that are related to this manual.

Contents

This chapter includes the following sections:

Topic	Page
Objectives of the Assessment Practices Manual	x
SNAP Beta Test	x
Intended Audience	x
Organization of the Assessment Practices Manual	x
Manual Revision Process	xii
Frequency of Changes	xii
Change Process	xii
Related IFPUG Documentation	xiv
Training Requirements	xvi

(cc) BY-NC-SA

Objectives of the Assessment Practices Manual

The primary objectives of the IFPUG Assessment Practices Manual are to:

- provide a clear and detailed description of Non-functional Assessment;
- ensure that assessments are consistent;
- provide guidance to allow non-functional assessment to be applied to popular methodologies and techniques; and
- provide a common understanding to allow tool vendors to provide automated support for non-functional assessment.

SNAP Beta Test

The purpose of the SNAP beta test was to repeat the spirit of Dr. Allan Albrecht's test of the initial version of the function point methodology, as documented in his 1977 paper "Measuring Application Development Productivity."[1] Our beta test, similar to Dr. Albrecht's function point test, found a statistically significant correlation between SNAP count size and work effort using a statistically large sample size of 48 applications containing over 500 data entries. The r^2 for the correlation between SNAP count and work effort was .89, the Spearman rank correlation was .85, the corresponding p-values for both tests was below .0001, and the test for randomness in the regression model passed the runs test.

These statistics mean that for this beta test, SNAP size was 89% of the reason for the work effort expended (the other 11% may result from different software languages, teams skills, counting errors, etc.)

Intended Audience

The standard in this manual should be applied by anyone to assess non-functional product size. The manual was designed for use by persons new to assessment as well as those who were trained and users with intermediate and advanced experience.

Organization of the Assessment Practices Manual

There are three major parts in the Assessment Practices Manual (APM)

- Part 1: The SNAP Method
- Part 2: Examples
- Part 3: Appendices

[1] A. J. Albrecht, "Measuring Application Development Productivity" Proceedings of the Joint SHARE, GUIDE, and IBM Application Development Symposium, Monterey, California, October 14–17, IBM Corporation (1979)

(cc) BY-NC-SA

Part 1 – The SNAP Method

Part 1 is the IFPUG non-functional size measurement method which contains the rules. To speak a language as a native, learning the grammar and the words alone are not sufficient. They just provide a framework. You need language experience to understand how the language is spoken in practice, how the grammar rules should be applied, what idiomatic expressions are common, and so on. The same is true for SNAP. The knowledge of process and rules, as reflected in Part 1, is a necessity, but the knowledge alone is not a sufficient condition to apply SNAP correctly. That's why the APM contains the parts below.

Part 2 – Examples

Part 2 provides detailed examples to explain counting practice concepts and rules. Each example should be considered on its own merits. Since each example is intended to illustrate a specific scenario, variations may exist between examples. Although the examples throughout the manual deal with similar subject matter, they are not intended to represent a single set of requirements.

Part 3 – Appendices

Part 3 contains additional information (Glossary, APM link to IFPUG CPM, Index and APM versions release notes).

In principle, each part stands on its own.

 (cc) BY-NC-SA

Manual Revision Process

This section explains the frequency of changes to the Assessment Practices Manual and defines the change process.

Frequency of Changes

During January of each year, a new version of the Assessment Practices Manual may become effective. It will include any new or changed definitions, rules, or Assessment practices that have been finalized by the Non-Functional Sizing Standards Committee (NFSSC) since the previous version.

Change Process

The following activities outline the process for adding or changing information in the Assessment Practices Manual. Explanations of each activity follow the table.

Step	Action
1	The issue is submitted to the NFSSC.
2	The issue is assigned for research.
3	The NFSSC reviews and discusses the issue.
4	The NFSSC presents a proposed solution to the IFPUG membership.
5	An impact study is initiated if the proposed change would have any impact on existing counts.
6	The final decision is made.
7	The IFPUG membership is informed of the decision.
8	Changes become effective with, and are reflected in, the next release of the Assessment Practices Manual.

1 Issue Submitted

The reader submits ideas, changes, or issues to the Non-Functional Sizing Standards Committee by sending email to ifpug@ifpug.org or nfssc@ifpug.org

2 Research Assigned

A member of the NFSSC is assigned the responsibility for identifying all alternatives, the rationale, and the potential impact of each alternative if it is implemented. Thorough examination of existing counting standards and historical papers is completed while compiling alternatives. In addition, an effort is made to determine what is thought to be common practice.

3 NFSSC Review

The NFSSC reviews and discusses the rationale for each alternative, and its potential impact. The review and discussion may result in a proposal for change or the review may lead the committee to reject the change request.

 (cc) BY-NC-SA

4 **Solution Proposed**	A proposed solution is made to the IFPUG membership and written comments are solicited. A copy of the proposed changes is mailed to IFPUG contacts at member organizations. The proposal also may be announced and distributed during an IFPUG conference. The latter depends on the timing of the committee meeting rather than the conference schedule.
5 **Impact Study Initiated**	The NFSSC has adopted a conservative stance on initiating impact studies. If it is possible that common practice must change, or several organizations or types of applications will be impacted by the change, an impact study is initiated. The success of the impact study is the responsibility of every IFPUG member. If the NFSSC receives written feedback indicating there is little or no impact, the study is discontinued.
6 **Final Decision Made**	The committee makes a final decision using results from research, written comments from members, and the impact study. The committee can complete more than one iteration of Steps 2 through 5 (research through impact study) before making a final decision. The final decision can result in a change or the committee may decide that a change is not warranted.
7 **Decision Communicated**	The final decision is communicated in writing to IFPUG members via the IFPUG contact at the various organizations. If any impact study results contributed to making a decision, the results and a recommendation on how to minimize the impact of the change will also be communicated.
8 **Decision Effective Date**	The Assessment Practices Manual will be updated to reflect the decisions. The effective date of the decisions is the date of the next January release of the manual.

(cc) BY-NC-SA

Related IFPUG Documentation

This Assessment Practices Manual is one module in the IFPUG documentation. All documents complement each other.

The following table describes related IFPUG publications.

Document	Description
IFPUG Brochure (Available)	This publication is an introduction to the International Function Point Users Group. It includes a brief history of the organization, introduces function point analysis, and defines the purpose of IFPUG. The brochure also includes a membership application. Audience: This publication is for anyone who wants an overview of IFPUG or an application for membership.
IFPUG: Organizational Structure and Services (Available)	This publication describes IFPUG services, and lists the board of directors, committees, and affiliate members worldwide. Audience: This publication is for anyone who wants background information about IFPUG.
Counting Practices Manual (Release Date: January 2010)	International Function Point Users Group (IFPUG) Function Point Counting Practices Manual (CPM), Release 4.3.1
Guidelines to Software Measurement (Release Date: August 2004)	This manual provides an overview of software metrics for organizations working to create or improve software measurement programs. The manual addresses both system and customer management, provides high-level justifications for software measurement, and examines the components of effective measurement programs. Audience: This manual is intended for IFPUG members, Function Point Coordinators, persons who prepare the reports to management, and other persons knowledgeable about and working directly with function points.
Quick Reference Counting Guide v2.0 (Release Date: 2009)	This quick reference guide is a summary of function point counting rules and procedures. Audience: This summary information is intended for anyone applying function point analysis.
Adjusted Functional Size Quick Reference Guide (Release Date: 2010)	This quick reference guide is a summary of the General Systems Characteristics. Audience: This is intended for anyone using the optional General Systems Characteristics.
IFPUG Glossary (Available with CPM and Guidelines for Software Measurement)	This is a comprehensive glossary that defines terms used across IFPUG publications. Audience: The glossary is recommended for anyone who receives any of the other IFPUG documents or anyone who needs definitions of IFPUG terms.
A Framework for Functional Sizing, IFPUG, September 2003	This paper explains that product size contains three dimensions: functional size, technical size and quality size. The IFPUG FPA-method provides a measure for the functional size.

(cc) BY-NC-SA

IT Measurement: Practical Advice from the Experts, Addison-Wesley, April 2002	This book is an excellent compilation of articles written by experts in the field of Information Technology. It was compiled by IFPUG to include recent insights in the application of software metrics in practice.
The IFPUG Guide to IT and Software Measurements (Release Date: 2012)	The IFPUG Guide to IT and Software Measurements brings together 52 leading software measurements experts from 13 different countries who share their insight and expertise. Covering measurement programs, function points in measurement, new technologies, and metric analysis, this volume: • Illustrates software measurement's role in new and emerging technologies • Addresses the impact of agile development on software measurements • Presents measurements as a powerful tool for auditing and accountability Include metrics for the CIO
Considerations for Counting with Multiple Media White Paper, Release 1.1, April 15, 2010	This white paper describes the two approaches currently being used by Certified Function Point Specialists when identifying unique functions in projects and applications where the same functionality is delivered on multiple media.
Function Point Analysis Case Studies (Release Dates: Case Study 1 Release 3.0 September 2005 (CPM 4.2) Case Study 2 Release 3.0 March 2006 (CPM 4.2) Case Study 3 Release 2.0:September 2001 (CPM 4.1) Case Study 4 Release 2.0 September 2005 (CPM 4.2)	The case studies illustrate the major counting techniques that comprise the Function Point Counting Practices Manual. The cases illustrate function point counts for a sample application. The cases include the counting that occurs at the end of the analysis phase of software development and after system construction. Audience: The case studies are intended for persons new to function point analysis as well as those with intermediate and advanced experience.
Metric Views, Volume 6 Issue 2, August 2012 Metric Views, Volume 8 Issue 1, February 2014 Metric Views, Volume 8 Issue 1, February 2014	The Next Frontier: Measuring and Evaluating the Non-Functional Productivity Using SNAP for FUI Creation and Enhancement Experience of a SNAP user
APM 2.1.3 Quick Reference Guide (QRG) - Spanish, Portuguese (2014); APM 2.3.0 QRG - updated to APM 2.3, English (2015)	These quick reference guides are a summary of SNAP counting rules and procedures.
SNAP Case Study 1	A SNAP case study that focuses on sizing a requirement under sub-category 1.2 Logical and Mathematical Operations
SNAP Case Study 2	How to Use Function Points and SNAP to Improve a Software Acquisitions Contract

 (cc) BY-NC-SA

Training Requirements

Usability evaluations of this publication have verified that reading the Assessment Practices Manual alone is not sufficient training to apply the process at the optimum level. Training is recommended, particularly for those new to SNAP.

A workshop of two days has been developed to provide training in SNAP by IFPUG.

IFPUG has developed an official training certification and a training path for SNAP.

SNAP Certification

The CSP (Certified SNAP Practitioner) exam, provided by IFPUG, is a test of both the knowledge of the counting rules laid out in the APM and the ability to apply those rules. The exam consists of two sections: Definition and Implementation. An individual must have at least 80% overall score in order to receive the CSP designation. They must also have at least 70% in each of the Definition and the Implementation sections of the Exam.

See http://www.ifpug.org/certification/csp-certification/ for more details.

(cc) BY-NC-SA

Part 1 –The SNAP Method

 (cc) BY-NC-SA

This page intentionally left blank

(cc) BY-NC-SA

Introduction to SNAP

Introduction This chapter presents the assessment framework for non-functional sizing and a description of the objectives and benefits of SNAP.

Contents This chapter includes the following sections:

Topic	Page
A Framework for Non-Functional Sizing	1-2
Relationship between APM and Other Standards	1-2
ISO/IEC/IEEE – Definitions	1-2
Project Effort	1-8
SNAP Framework	1-9
SNAP Objectives and Benefits	1-10

 (cc) BY-NC-SA

A Framework for Non-Functional Sizing

The main objective of IFPUG's Framework for Non-Functional Sizing (2008) project was to ensure that a non-functional framework can be used to establish a link between non-functional size and the effort to provide the non-functional requirements.

The resulting framework has the following characteristics:

- The overall framework is an assessment of the size of non-functional requirements.
- The framework is comprised of assessment categories and sub-categories.
- Sub-categories are evaluated using specified criteria.
- The evaluation utilizes both assessed and/or measured criteria.

The non-functional assessment results have the following characteristics:

- They can be used in conjunction with the functional size, and will help explain the variance in development effort and productivity.
- Along with functional size, they can be used as input to estimating models.
- They are determined from the users' non-functional view, but understood and agreed by the users and by the development teams.

Relationship between APM and Other Standards

IFPUG's Assessment Practices Manual (APM) for the Software Non-Functional Assessment Process (SNAP) uses definitions and terminology from relevant international standards organizations such as ISO, IEC and IEEE wherever possible.

A set of base definitions is given in the following section, regarding the classification of requirements.

(cc) BY-NC-SA

ISO/IEC/IEEE – Definitions

User Requirements

In 1998, the first ISO/IEC Functional Size Measurement standard was published (ISO/IEC 14143-1:1998 "Software and Systems Engineering – Software measurement – Functional size measurement – Definition of concepts"). This standard defines the Functional Size as "a size of the software derived by quantifying the Functional User Requirements" (FUR). (This standard was updated in 2007 and is currently published as ISO/IEC 14143-1:2007.)

ISO/IEC 14143-1 distinguishes two subsets of user requirements (UR):

Functional User Requirements

Non-Functional Requirements (NFR)

ISO/IEC 9126-1:2001 provided the definition of the characteristics and associated quality evaluation process to be used when specifying the requirements for and evaluating the quality of software products throughout their life cycle.

ISO/IEC 25010:2011 has replaced and improved ISO/IEC 9126-1

ISO/IEC 14143-1 definition of FUR is as follows:

Functional User Requirements

"A subset of the User Requirements (UR). Requirements that describe what the software shall do, in terms of tasks and services."

Note: Functional User Requirements include but are not limited to:

- data transfer (for example, input customer data, send control signal),
- data transformation (for example, calculate bank interest, derive average temperature),
- data storage (for example, store customer order, record ambient temperature over time), and
- data retrieval (for example, list current employees, retrieve aircraft position).

Examples of User Requirements that are not Functional User Requirements include but are not limited to:

- quality constraints (for example, usability, reliability, efficiency and portability),
- organizational constraints (for example, locations for operation, target hardware and compliance to standards),
- environmental constraints (for example, interoperability, security, privacy and safety), and
- implementation constraints (for example, development language, delivery schedule).

ISO/IEC 9126-1:2001 defines functionality as follows:

Functionality is the set of attributes that bear on the existence of a set of

Functions and their specified properties. The functions are those that satisfy stated or implied needs.

 (cc) BY-NC-SA

ISO/IEC 14143-1 does not provide a definition for non-functional requirements themselves, but gives some examples in a note after FUR definition. In 2009, a separate initiative under development (ref. ISO/IEC/IEEE 24765:2010 Systems and software engineering – Vocabulary) proposed a formal definition of a non-functional User Requirement, as follows:

Non-Functional User Requirements (NFR)

A software requirement that describes not what the software will do but how the software will do it. [ISO/IEC 24765, Systems and Software Engineering Vocabulary.] Syn: design constraints, non-functional requirement. See also: functional requirement.

Examples include: software performance requirements, software external interface requirements, software design constraints, and software quality attributes.

Note: Non-functional requirements are sometimes difficult to test, so they are usually evaluated subjectively.

Again and analogously, ISO/IEC/IEEE 24765 defines a Functional Requirement as:

- *A statement that identifies what a product or process must accomplish to produce required behavior and/or results. [ISO/IEC 26710:2007 (IEEE std. 1220-2005) IEEE Standard for the Application and Management of the Systems Engineering Process. § 3.1.1.6.]*
- *A requirement that specifies a function that a system or system component must be able to perform. [ISO/IEC 24765, Systems and Software Engineering Vocabulary.]*

ISO/IEC 25010:2011

The product quality model categorizes product quality properties into eight characteristics (functional suitability, reliability, performance efficiency, usability, security, compatibility, maintainability and portability).

Each characteristic is composed of a set of related sub-characteristics.

1. **Functional Suitability**

   Degree to which a product or system provides functions that meet stated and implied needs when used under specified conditions

   1.1. Functional Completeness

   1.2. Functional Correctness

   1.3. Functional Appropriateness

2. **Performance Efficiency**

   Performance relative to the amount of resources used under stated conditions.

   2.1. Time behavior

   2.2. Resource utilization

   2.3. Capacity

 (cc) BY-NC-SA

3. Compatibility

Degree to which a product, system or component can exchange information with other products, systems or components, and/or perform its required functions, while sharing the same hardware or software environment.

3.1. Co-existence

3.2. Interoperability

4. Usability

Degree to which a product or system can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use.

4.1. Appropriateness recognizability

4.2. Learnability

4.3. Operability

4.4. User error protection

4.5. User interface aesthetics

4.6. Accessibility

5. Reliability

Degree to which a system, product or component performs specified functions under specified conditions for a specified period of time.

5.1. Maturity

5.2. Availability

5.3. Fault tolerance

5.4. Recoverability

6. Security

Degree to which a product or system protects information and data so that persons or other products or systems have the degree of data access appropriate to their types and levels of authorization.

6.1. Confidentiality

6.2. Integrity

6.3. Non-repudiation

6.4. Accountability

6.5. Authenticity

7. Maintainability

Degree of effectiveness and efficiency with which a product or system can be modified by the intended maintainers.

7.1. Modularity

7.2. Reusability

7.3. Analyzability

7.4. Modifiability

7.5. Testability

 (cc) BY-NC-SA

8. Portability

Degree of effectiveness and efficiency with which a system, product or component can be transferred from one hardware, software or other operational or usage environment to another.

8.1. Adaptability

8.2. Installability

8.3. Replaceability

Relationship between APM and ISO/IEC 9126-1, ISO/IEC 25010

SNAP sub-categories do not define or describe non-functional requirements; they classify how these requirements are met within the software product.

A nonfunctional requirement, which is defined either by ISO/IEC 9126-1 or ISO/IEC 25010, may be implemented in the product by using more than one sub-category. Accordingly, a sub-category may cater for several quality characteristics, as defined either by ISO/IEC 9126-1 or ISO/IEC 25010.

A view of the relationship is presented in the example and in table 1-1 below.

Example

This example analyses a requirement to improve the way system recovers from a crash:

Using ISO/IEC 25010, this requirement falls under the area of Reliability, and the attribute is Recoverability

SNAP sizes this requirement according to the design:

- An algorithm is added to identify corrupted data in specific fields.
- Time stamps are added to database records.
- An algorithm is written to reconstruct corrupted data using uncorrupted record.

The design involves the following SNAP sub-categories:

- Database Technology (adding time stamp).
- Logical and Mathematical operations.

In this example, the "recoverability" type of requirement is mapped to two sub-categories, "Database Technology" and "Logical and mathematical operations.

More examples of such mapping are illustrated in table 1-1.

Note:

All examples in table 1-1 assume that these requirements are not covered by Function Points.

To learn more about the link between FPA and SNAP, please refer to Appendix B and tables 3-1 and 3-2

		SNAP Categories and sub-categories														
		Data Operations					Interface Design				Technical Environment			Architecture		
ISO 25010 Sub-characteristics (A partial list for demonstration)		Data entry validations	Logical and Mathematical operations	Data Formatting	Internal Data Movements	Delivering Added value to users by Data	User Interfaces	Help Methods	Multiple Input Methods	Multiple Output Methods	Multiple platforms	Database Technology	Batch Processes	Component based software	Multiple Input/output Interfaces
		1.1	1.2	1.3	1.4	1.5	2.1	2.2	2.3	2.4	3.1	3.2	3.3	4.1	4.2
Portability	Replaceability														
	Installability														
	Adaptability														
Security	Changeability														
	Authenticity	**Ex. 2**													
Reliability	Fault tolerance														
	Recoverability														
Usability	Accessibility														
	User error protection														
	Operability								**Ex. 4**						
	User interface aesthetics						**Ex. 1**								
	Learnability														
	Appropriateness recognizability														
Performance Efficiency	Capacity														
	Resource utilization														
	Time behavior											**Ex. 3**			
Functional Suitability	Functional Completeness														
	Functional Appropriateness														
	Functional Correctness														

Table 1-1 Mapping ISO/IEC 25010 characteristics to SNAP sub categories

The example numbers in Table 1-1 refer to the following:

Ex. 1. Improving the understandability and learnability by adding pop-up help menus (Sub-category 2.1) and rearranging the screens (Sub-category 2.2)

Ex. 2. Improving Security by adding more validations to authentication process, using Sub-category 1.1 “data entry validations” and Sub-category 1.2 “Logical and Mathematical operations”

Ex. 3. Improving performance by adding indices to the database and improving queries (Sub-category 3.2)

Ex. 4. Adding Barcode reading as additional input method (Sub-category 2.3)

 (cc) BY-NC-SA

Project Effort

Software Project

A collaborative enterprise, which is planned and executed to achieve a particular aim mainly by means of software development.

ISO/IEC/IEEE 16326:2009 "Systems and Software engineering – lifecycle processes – Project management" (Formerly IEEE Std-1058-1998, "IEEE Standard for Software Project Management Plans"): The set of work activities, both technical and managerial, required to satisfy the terms and conditions of a project agreement

Project effort is invested to provide the following:

- Features that are built into the product to meet the functional requirements (See definition in previous chapter);
- Features that are built into the product to meet the non-functional requirements (See definition in previous chapter); and
- Project-related tasks, to ensure that:
  - The project is managed;
  - The project meets its quality, timeline and budget constraints; and
  - Risks are managed.

Meeting functional requirements (FUR) – the Product functional size

The functional size could be measured/assessed using the IFPUG functional size measurement method, based on the FURs.

Meeting non-functional requirements (NFR) –the Product non-functional size

The SNAP framework provides the basis for sizing non-functional requirements.

Performing project-related tasks

Project related tasks do not affect the Product size. Although these tasks affect the effort required to deliver the product, they influence the productivity and not the software size.

Examples of project related tasks:

- Team training;
- User training; and
- Project documentation (such as manuals, plans, status reports, roadmaps, work instructions, quality standards).

Function Points Analysis (FPA) and SNAP result in different size-measures representing different dimensions of product size. While these sizes cannot be added together because they represent different dimensions (like volume and temperature of a room), they can be used together in estimating the effort towards the development of an application or a system.

In addition, project related tasks are also used to estimate the effort to develop the application or system.

 (cc) BY-NC-SA

SNAP Framework

The SNAP framework and FPA can be seen as three dimensions of a block:

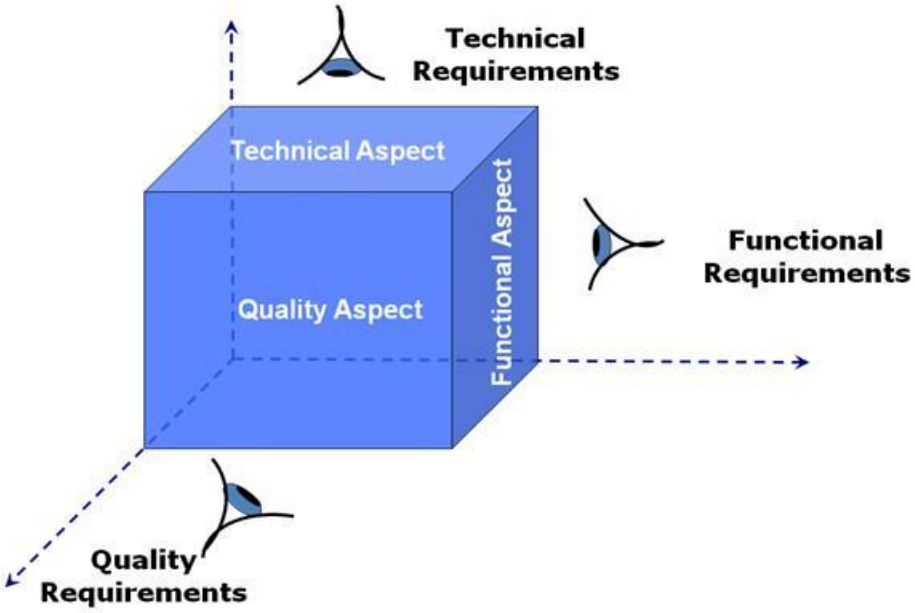


Figure 1-1 The Framework and Requirements

The dimension for the functional requirements (FUR) and the functional perspective of the software development project are currently covered by the functional size measure - IFPUG function points as defined in the IFPUG Counting Practice Manual. The dimensions of the non-functional requirements (both technical and quality) and the non-functional perspectives of the software development project are defined in this manual.

The following are the standards organizations definitions of technical requirements:

- ISO – requirements relating to the technology and environment, for the development, maintenance, support, and execution of the software.
- IEEE – combination of design, implementation, interface, performance, and physical requirements.

The Quality Requirements are requirements that are not defined as functional or technical, and relate to the quality of the system or components. The following are available definitions:

- ISO – The following characteristics form part of the quality model: Functionality, Reliability, Usability, Efficiency, Maintainability, and Portability
- IFPUG – Quality includes: conformity to user expectations, user requirements, customer satisfaction, reliability, and level of defects present.

 (cc) BY-NC-SA

SNAP Objectives and Benefits

Objectives

SNAP measures software by quantifying the size of non-functional requirements. With this in mind, the objectives of SNAP are to:

- Measure the non-functional size of the software that the user requests and receives.
- Demonstrate the full economic value of the application, including its functional aspects as well as the non-functional aspects (have the non-functional baseline as well as the functional baseline, to demonstrate the full economic value).
- Measure software development and maintenance projects based on the non-functional requirements.
- Size technical projects, in which FPA is not applicable.

In addition to meeting the above objectives, the process of assessing non-functional requirements should be:

- Simple enough to minimize the overhead of the measurement process.
- A consistent measure among various projects and organizations. SNAP allows to determine (by counting each of the four categories, from each one of the sub-characteristics) the possibility to size and therefore better estimate a project with/without FPs, according to the set of user requirements received for a project.

(cc) BY-NC-SA

Benefits

A non-functional assessment will assist IT organizations in multiple ways. It will provide insight into the delivery of projects and maintenance of applications to assist in estimating and in the analysis of quality and productivity. Used in conjunction with FP measures, the non-functional assessment will provide information that can identify items impacting quality and productivity in a positive or negative way.

Having this information enables software professionals to:

- better plan and estimate projects,
- identify areas of process improvement,
- assist in determining future non-functional strategies,
- quantify the impacts of the current non-functional strategies, and
- Provide specific data when communicating non-functional issues to various audiences.

Organizations can apply SNAP as:

- a methodology to measure the non-functional size of a software product to support quality and productivity analysis,
- a methodology to estimate cost and resources required for software development and maintenance,
- a methodology to measure cost reduction for software development and maintenance, in addition to FPA,
- a normalization factor for software comparison,
- a methodology to determine the non-functional size of a purchased application package by assessing all the portions and categories included in the package, and
- a methodology to help users determine the benefit of an application package to their organization by assessing portions or categories that specifically match their requirements.

Note:

"Function Points + SNAP points" are **not** equal to the overall product size.

As of the date of this publication, the size of a software application is considered to have two distinct parts: the size of the FURs and the size of the NFRs. For example, if an application's functional size is 700 function points and non-functional size is 200 SNAP points, then the entire size could be stated as "700 function points, and 200 SNAP points." The two sizes do not sum up. ."

The IFPUG functional sizing methodology does not change when measuring the non-functional requirements using SNAP.

A project may have 0 Function Points and non-zero number of SNAP points, or 0 SNAP Points and non-zero number of Function Points, or any combination of Function Points and SNAP Points

Further research is needed to determine if function points and SNAP points can be somehow combined as part of single metric.

 (cc) BY-NC-SA

This page intentionally left blank

 (cc) BY-NC-SA

SNAP Overview

Introduction This chapter presents an overview of SNAP. It presents a summary of the related concepts and process steps.

Contents This chapter includes the following:

Topic	Page
SNAP Description	2-2
Non-Functional Assessment Process	2-2
Section 1: Determine Assessment Purpose, Scope, Boundary and Partition	2-3
Section 2: Associate Non-functional Requirements with Categories and Sub-categories	2-5
Section 3: Identify the SNAP Counting Units (SCUs)	2-7
Section 4: Determine the Complexity of each SNAP Counting Unit (SCU)	2-7
Section 5: Determine the SNAP Points (SP) of Each SCU	2-8
Section 6: Calculate Non-functional Size	2-8

 (cc) BY-NC-SA

SNAP Description

Introduction

The purpose of this chapter is to describe how non-functional requirements can be sized using SNAP method.

Looking at the various scenarios on non-functional aspects within an application, various sub-categories are identified and are grouped under logical categories.

The categories and sub-categories do not replace or explain the standards that describe and classify the non-functional requirements (such as ISO/IEC 25010:2011). The categories and sub-categories describe how the assessed project or product will meet these non-functional requirements.

Non-Functional Assessment Process

The non-functional assessment will use a series of questions grouped by sub-categories to measure the size of non-functional requirements for the development and delivery of the software product.

- The categories will focus on those non-functional requirements that affect product size
- The process will allow for the sizing of the non-functional requirements using a series of questions and measures.
- The process can be used for Development Projects, Enhancement Projects, Maintenance Activities, and Applications.

Procedure by Section

The following table shows the SNAP procedure as explained in the remaining chapters of Part 1.

Procedure	Section
Determine Assessment Purpose, Scope, Boundary and Partition	1
Associate non-functional requirements with categories and sub-categories	2
Identify the SNAP Counting Units (SCUs)	3
Determine the complexity of Each SNAP Counting Unit (SCU)	4
Determine the SNAP Points (SP) of each SCU	5
Calculate Non-Functional Size (SNAP Points)	6

 BY-NC-SA

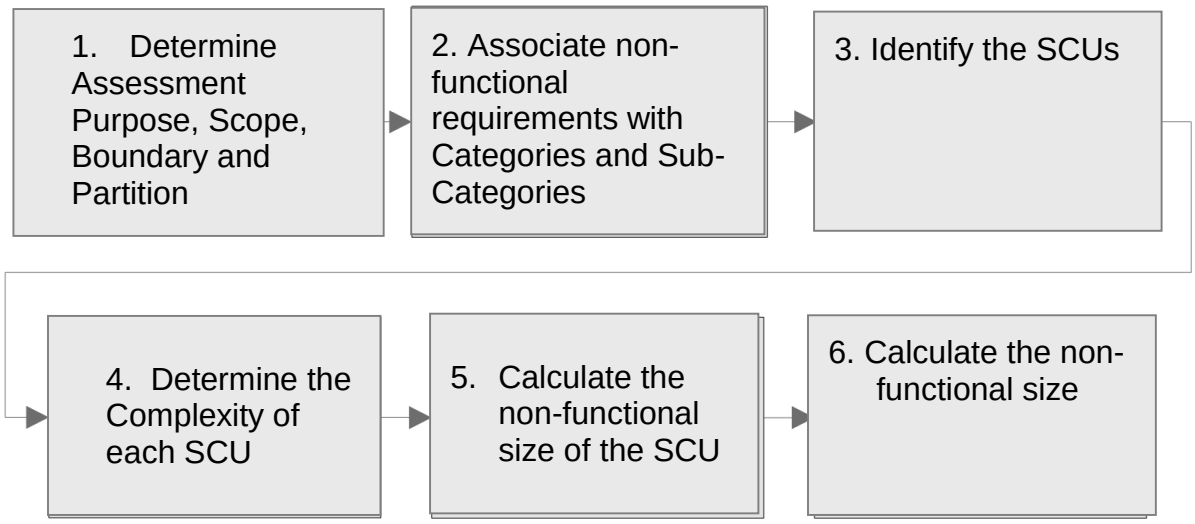

1. Determine Assessment Purpose, Scope, Boundary and Partition
2. Associate non-functional requirements with Categories and Sub-Categories
3. Identify the SCUs
4. Determine the Complexity of each SCU
5. Calculate the non-functional size of the SCU
6. Calculate the non-functional size

Figure 1-2 SNAP Diagram

Section 1: Determine Assessment Purpose, Scope, Boundary and Partition

The following steps shall be performed when identifying the Assessment Purpose, Scope, Boundary and Partition:

Step	Action
1	Identify the purpose of the assessment
2	Identify the assessment type
3	Identify the assessment scope
4	Identify the boundary of the application(s)
5	Identify the partitions, if applicable
6	Document purpose, type, scope, boundary, partition, and assumptions

See Chapter 4 for more details about the purpose, scope, and boundary of the assessment.

This section defines the assessment type, assessment scope, boundaries and partition

Assessment Type

The functional size and the non-functional size can be measured for either projects or applications. The type of assessment is determined, based on the purpose, as one of the following:

- Development project assessment.
- Enhancement project assessment.
- Application assessment.

Chapter 4 in Part 1 defines each type of assessment.

 (cc) BY-NC-SA

Boundary

The boundary is a conceptual interface between the software under study and its users.

The boundary (also referred to as "application boundary"):

- Defines what is external to the application.
- Indicates the border between the software being measured and the user
- Acts as a "membrane" through which data processed by transactions pass into and out of the application.
- Is dependent on the user's external business view of the application; it is independent of non-functional and/or implementation considerations.

Partition

A partition is a set of software functions within an application boundary that share homogeneous assessment criteria and values. A partition requires development effort, that may not be reflected when sizing the functional aspect of the project/product, using FPA

The positioning of the partition may be subjective. It is often difficult to delineate where one partition stops and another begins. Try to place the partition from a non-functional perspective of the users, such as maintainability, portability, or installability, rather than based on technical or physical considerations. It is important that the partition is placed with care, since all data crossing the partition impact SNAP size.

- The partition is determined based on the user view. The focus is on what the user can understand and describe.

Within a boundary, partitions:

- contain all the software functions which constitute the overall functionality of the application being assessed;
- may cooperate between themselves in order to provide complete software functions to the application user;
- shall not overlap; and
- shall be consistent over time.

A partition:

- may be used to meet non-functional requirements;
- can be sized using SNAP categories and sub-categories; and
- might coincide with the overall application (such as: client and server residing in a single system).

In case where there are no identifiable partitions the boundary itself is taken, and no partitions are considered.

Examples of partitions

Examples of partitions are:

- the client functions in a client-server application,
- the server functions in a client-server application,
- the functions of "user A", (to be) separately designed and/or implemented from functions of "user B", within the same application,
- the functions (to be) implemented over non-functional platform "X",

 BY-NC-SA

separately identified from functions (to be) implemented over non-functional platform "Y", within the same application,

- SOA – Application within the boundary, and
- peer component within the boundary.

Note: Counting SNAP points is performed at boundary level. Partitions add SNAP points by using sub-category 1.4 Internal Data Movements (See Figure 1-3)

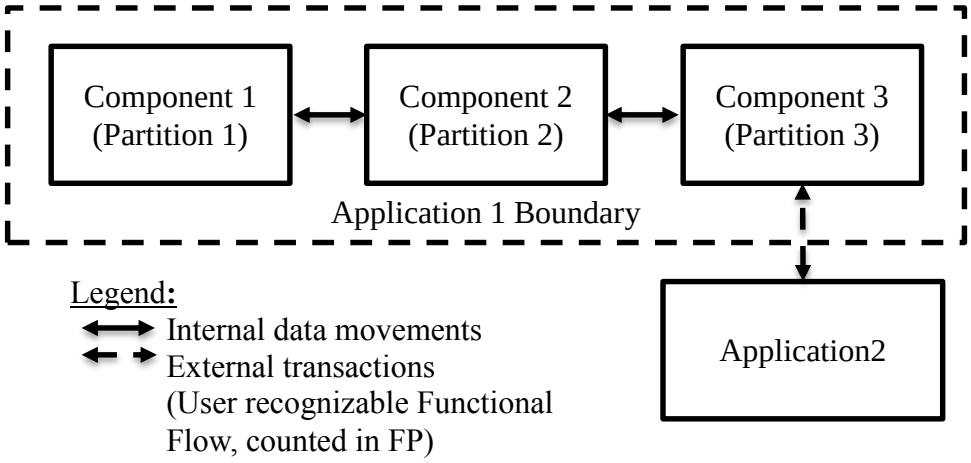


Figure 1-3 Relations between partitions and applications

Section 2: Associate Non-functional Requirements with Categories and Sub-categories

This section defines the categories and sub-categories, and describes the association of the non-functional requirements with sub-categories. The categories and sub-categories are standard for any SNAP assessments.

Category Definition A category is a group of components, processes or activities that are used in order to meet the non-functional requirement.

- Categories classify the non-functional requirements.
- Categories are generic enough to allow for future technologies.
- Categories are divided into sub-categories. Each sub-category has common features (within the sub-category). This simplifies the assessment.

Each SNAP category groups the sub-categories based on the same level of operations and/or similar type of activities executed by the non-functional assessment process.

 BY-NC-SA

Sub-category Definition

Sub-category is defined as a component, a process or an activity executed within the project, to meet the non-functional requirement.

Note: A non-functional process may have to execute more than one sub-category to meet the non-functional requirement.

Category & Sub-category

The association of the non-functional requirements with categories and sub-categories is performed as follows:

- Identify the non-functional requirement under scope (for example, requirements for data security; requirements to improve performance)
- Analyze the design and identify which sub-categories are used in order to meet the requirement

Categories and sub-categories are:

1. Data Operations
   - 1.1. Data Entry Validations
   - 1.2. Logical and Mathematical Operations
   - 1.3. Data Formatting
   - 1.4. Internal Data Movements
   - 1.5. Delivering Added Value to Users by Data Configuration
2. Interface Design
   - 2.1. User Interfaces
   - 2.2. Help Methods
   - 2.3. Multiple Input Methods
   - 2.4. Multiple Output Methods
3. Technical Environment
   - 3.1. Multiple Platforms
   - 3.2. Database Technology
   - 3.3. Batch Processes
4. Architecture
   - 4.1. Component Based Software
   - 4.2. Multiple Input / Output Interfaces

The sub-categories address the non-functional requirements, including Technical and Quality requirements. (Quality requirements such as usability or reliability, as defined by ISO/IEC 9126-1 or ISO/IEC 25010, can be addressed by the following sub-categories: Data Entry Validation, Data Formatting, and User Interface. For example, adding On-line Help is a requirement to improve ease of learning and can be sized by using the sub-category "Help Methods.")

See Chapter 5 for more details about the categories and sub-categories.

 (cc) BY-NC-SA

Section 3: Identify the SNAP Counting Units (SCUs)

This section defines the rules and procedures that apply when identifying the SNAP Counting Units (SCUs).

The SCUs are unique to each sub-category; they are determined by the nature of the sub-category.

The SCU is part of the sub-category definition.

Sizing is done separately per each SCU.

Note: A requirement may contain both functional and non-functional aspects. In such a case, the requirement will have a functional size, measured in Function Points, and a SNAP size, measured in SNAP points.

Use Part 3, Appendix B ("Counting Function Points and SNAP points") for requirements that involve both functional and non-functional requirements.

The example in Part 2 demonstrates how SCUs are used.

SNAP Counting Unit (SCU) Definition

The SCU is a component or activity, in which complexity and size is assessed.

The SCU can be a component, a process or an activity identified according to the nature of the sub-category/sub-categories.

An SCU may contain both functional and non-functional characteristics. In these cases, sizing of the elementary process will be performed for its functional sizing, using function point analysis, and for its non-functional sizing, using SNAP.

See Chapter 5 for more details about the SCU.

Section 4: Determine the Complexity of each SNAP Counting Unit (SCU)

This section defines how to determine the complexity and size of each SCU within the sub-category.

Answer the assessment questions for each sub-category.

The assessment questions are related to parameters that affect the complexity of a given sub-category.

The assessment rating is the answer to the assessment questions.

The complexity level of an assessment rating or the value of the parameters within each SCU is mapped to a size.

See Chapter 5 for more details about the determination of complexity.

 BY-NC-SA

Section 5: Determine the SNAP Points (SP) of Each SCU

This section defines how to determine the size of each sub-category.

SNAP Points Definition

SNAP points (SP) are the sum of the size of all SCUs identified in each sub-category.

Once all complexity parameters have been assessed, the size of each SCU is calculated and the SNAP Points of all SCUs are added together to obtain the calculated SNAP Points for the sub-category.

See Chapter 5 for more details how to calculate the SNAP Points for each SCU.

Section 6: Calculate Non-functional Size

This section defines how to calculate the size of the non-functional aspect of the project/product in scope.

The SNAP Points are the final non-functional size obtained by combining all category values.

When more than one sub-category is identified, the overall non-functional size shall be obtained by combining the size of each sub-category within the application boundary of the software product being assessed.

Note:

When many (more than one) sub-categories are using the same definition of SCU, such as the elementary process, then we should answer the assessment questions from all relevant sub-categories for this elementary process.

Counting will be per SCU per sub-category. Therefore an SCU may generate SNAP Points from more than one sub-category. Requirement # 1 in the case study that is presented as an example.

See Chapter 6 for more details about the calculations and the definition of the formulas.

 (cc) BY-NC-SA

Assessment Preparation

Introduction This chapter presents the concept of the user's role in defining the non-functional requirements for a project or application. This chapter also includes a description of useful documentation during the life cycle of an application.

Contents This chapter includes the following:

Topic	Page
Timing of Non-functional Assessments	3-2
Useful Project/Application Documentation	3-3
Estimated and Final Non-functional Assessment	3-4

 (cc) BY-NC-SA

Timing of Non-functional Assessments

Non-functional assessments can be completed at any time in the development life cycle to aid in project estimating, monitoring project change of scope, and evaluating delivered non-functional requirements.

Prior to beginning a non-functional assessment, determine whether you are approximating or measuring the size and document any assumptions.

Approximating permits assumptions to be made about unknown non-functional categories and/or their complexity in order to determine an approximate non-functional size.

Measuring includes the identification of all applicable non-functional sub-categories and their complexity to accomplish a non-functional size analysis.

At an early stage non-functional requirements may not be fully defined. Despite the disadvantages, this assessment can be very useful to produce an early estimate. Uses of the non-functional assessment for approximating or measuring non-functional size at the various life cycle phases are presented below:

Life Cycle Phase	**SNAP Points can be approximated**	**SNAP Points can be measured**
Proposal: users express needs and intentions	Yes	No
Requirements: developers and users review and agree upon expression of user needs and intentions	Yes	No
Design: developers may include elements for implementation	Yes	Yes
Construction	Yes	Yes
Delivery	Yes	Yes
Maintenance (Adaptive – modifying the system to cope with changes in the software environment. Perfective – implementing new or changed user requirements which concern functional enhancements to the software. Preventive – increasing software maintainability or reliability to prevent problems in the future)	Yes	Yes
Maintenance (Corrective - Reactive modification of a software product performed after delivery to correct discovered problems)	SNAP Points not used	SNAP Points not used
Note: No specific development life cycle is implied. If using an iterative approach, you may expect to approximate/measure the SNAP size multiple times during the project life cycle.		

Table 1-2 Timing of SNAP Assessment

 BY-NC-SA

Useful Project/Application Documentation

Typically software development life cycle activities include the development of a Technical Requirements document and/or a Non Functional Requirements document. The Technical Requirements may include elements which are necessary for the implementation, but which are not considered in functional size measurement (e.g., temporary files, index, etc.).

This document may have one or more of the following characteristics:

- Technology dependence.

  For example, physical files vary based on the database environment.
- Terminology unfamiliar to the users

  For example, software developers may refer to physical files rather than to logical groups of data.
- Technical constraints.

  For example, the computing capacity (infrastructural support to aid processing) currently available in the organization.
- Physical boundaries.

  For example, there may be separate technical requirements for client and server

In general, the following items are useful when conducting any non-functional assessment:

- Requirements documents
- Entity relationship diagrams
- Technical requirements documents
- Object models
- Physical data models
- Physical file and database layouts
- Interface agreements with descriptions of batch feeds/transaction files and interfaces to/from other applications
- Samples of reports, displays, and other user interfaces
- Demonstration of application operation
- UI standards
- Availability of one or more technical experts will help in a great way (for the application being assessed)
- System design documentation
- Technical design document
- Architecture diagrams
- Use cases/functional requirement document

Note: The list above is not all-inclusive.

 (cc) BY-NC-SA

Estimated and Final Non-functional Assessment

It is important to realize that early non-functional assessments are estimates of to-be-delivered non-functional requirements. In addition, as the scope is clarified and the requirements developed / evolved, it is quite normal to identify additional non-functional characteristics, which were not specified in the original requirements. This phenomenon is sometimes called "scope creep." It is essential to update the application size upon completion of the project. If the size changes during development, then the non-functional size at the end of the life cycle should accurately reflect the full non-functional characteristics delivered to the user.

 BY-NC-SA

Determine Purpose, Scope, Boundary and Partition

Introduction This chapter describes in detail the steps in the determination of scope and boundary. It explains how they are influenced by the purpose of the assessment and the type of assessment.

Contents This chapter includes the following sections:

Topic	Page
Steps for Determination of Scope and Boundary	4-2
Identify the Purpose of the Assessment	4-2
Identify the Type of Assessments	4-3
Diagram of Types of Assessments	4-4
Identify the Scope of the Assessment	4-4
Determine the Boundary	4-5
Determine the Partition	4-6
Rules and Procedures	4-6

 (cc) BY-NC-SA

Steps for Determination of Scope and Boundary

When identifying Scope and Boundary, the following steps shall be performed:

1. Identify the purpose of the assessment.
2. Identify assessment type, based on the purpose.
3. Determine the assessment scope, based on the purpose and type of count.
4. Determine the boundary of the application(s).
5. Determine the partitions, if applicable.
6. Document the following items:
   - The purpose and type of the assessment,
   - The assessment scope,
   - The boundary of the application(s),
   - The partition(s) within the boundary(s), and
   - Any assumptions related to the above.

Consistency with FPA

The purpose, scope, and logical application boundaries need to be consistent between the FPA and SNAP process.

See Appendix B for the link between FPA and SNAP.

Identify the Purpose of the Assessment

A non-functional size measurement is conducted to measure the size of non-functional requirements for the development and delivery of a software product. In order to provide answers relevant to the purpose for the assessment, the assessment scope, boundaries, and partitions must be separately identified.

The purpose:

- Determines the type of non-functional assessment and the scope of the required assessment to obtain the non-functional size.
- Determine the size of the non-functional aspect of the assessed products.

Examples of purposes are to provide the non-functional size:

- of a development project as an input to the estimation process to determine the effort to develop the first release of an application.
- of the installed base of applications to determine the support costs.
- delivered by an enhancement.
- for maintenance activities.

 BY-NC-SA

Identify the Type of Assessments

The functional size and the non-functional size can be measured for either projects or applications. The type of assessment is determined, based on the purpose, as one of the following:

- Development project assessment.
- Enhancement project assessment.
- Application assessment.

The following paragraphs define each type of assessment.

Development Project

A **development project** is a project to develop and deliver the first release of a software application.

DSP – Development Project SNAP points.

The **development project non-functional size** is an assessment of the non-functional requirements provided to the users with the first release of the software, as measured by the development project SNAP assessment by the activity of applying the SNAP method.

Enhancement Project

An **enhancement project** is a project to develop and deliver corrective, preventive, adaptive or perfective maintenance.

ESP – Enhancement Project SNAP points.

The **enhancement project non-functional** size is a measure of the non-functional characteristics added, changed or deleted at the completion of an enhancement project, as measured by the enhancement project SNAP assessment.

Application

An **application** is a cohesive collection of automated procedures and data supporting a business objective; it consists of one or more components, modules, or sub-systems.

ASPA – The application SNAP Points after the enhancement project.

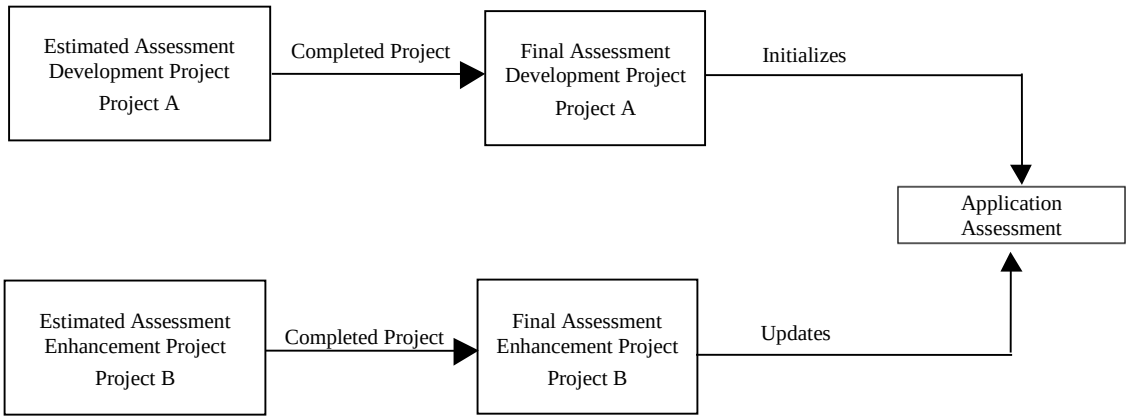
An **application's non-functional size** is a measure of the non-functional characteristics that an application provides to the user, determined by conducting the application SNAP assessment.

It is also referred to as the baseline or installed non-functional size. This size provides a measure of the current non-functional characteristics the application provides the user. This number is initialized when the development project SNAP assessment is completed. It is updated every time a completion of an enhancement project alters the application's non-functional size.

 (cc) BY-NC-SA

Diagram of Types of Assessments

The following diagram illustrates the types of SNAP assessments and their relationships. (Project A is completed first, followed by Project B.)



The previous example shows the non-functional assessment concept but does not reflect the full example of the process.

Identify the Scope of the Assessment

The assessment scope defines the set of non-functional user requirements to be included in the assessment. The scope:

- Is determined by the purpose for performing the non-functional assessment.
- Defines a set of partition(s).
- Identifies which non-functional assessment categories and sub-categories will be included in the non-functional size measurement to measure the size of non-functional requirements for the development and delivery of the software product.
- Could include more than one application.

The scope of:

- A development project non-functional assessment includes all non-functional requirements for the development and delivery of the software product.
- An assessment of an installed base of applications includes all non-functional requirements for the support of the installed applications.
- An enhancement non-functional assessment includes all non-functional requirements for the development and delivery of the enhancement project; the boundary of the application(s) impacted remains the same.
- A maintenance assessment includes all non-functional requirements for a selected scope.

(cc) BY-NC-SA

Determine the Boundary

Determine the boundary of each application within the assessment scope, based on the user view.

User view definition

In order to establish the boundary, the user view must be defined. The following hints can help you to identify the user view:

- A user is any person or thing (application, device, etc.) that communicates or interacts with the software at any time,
- A user view consists of the functional and non-functional requirements as perceived by the user, and
- A partition may act as an "internal user" for another partition within the same application boundary, in terms of data exchange or data sharing; consequently, different non-functional assessments might be created for each partition.

A user view:

- is a description of the business functions and non-functional requirements;
- represents a formal description of the user's needs in the user's language;
- can be verbal statements made by the user as to what their view is;
- is approved by the user;
- can be used to measure the functional and non-functional size; and
- can vary in physical form (e.g., catalog of transactions, proposals, requirements document, external specifications, detailed specifications, user handbook, quality or non-functional specifications).

Boundary definition

The boundary is a conceptual interface between the software under study and its users.

The boundary (also referred to as application boundary):

- Defines what is external to the application.
- Indicates the border between the software being measured and the user.
- Acts as a "membrane" through which data processed by transactions pass into and out of the application.
- Is dependent on the user's external business view of the application; it is independent of non-functional and/or implementation considerations.

The positioning of the boundary between the software under investigation and other software applications may be subjective. It is often difficult to delineate where one application stops and another begins. Try to place the boundary from a business perspective rather than based on technical or physical considerations.

 BY-NC-SA

For example, the following diagram shows boundaries between the Human Resources application and the external applications, Currency and Fixed Assets. The example also shows the boundary between the human user (User 1) and the Human Resources application. The Human Resource application may in turn internally satisfy the functional, technical and quality requirements specified by the user.

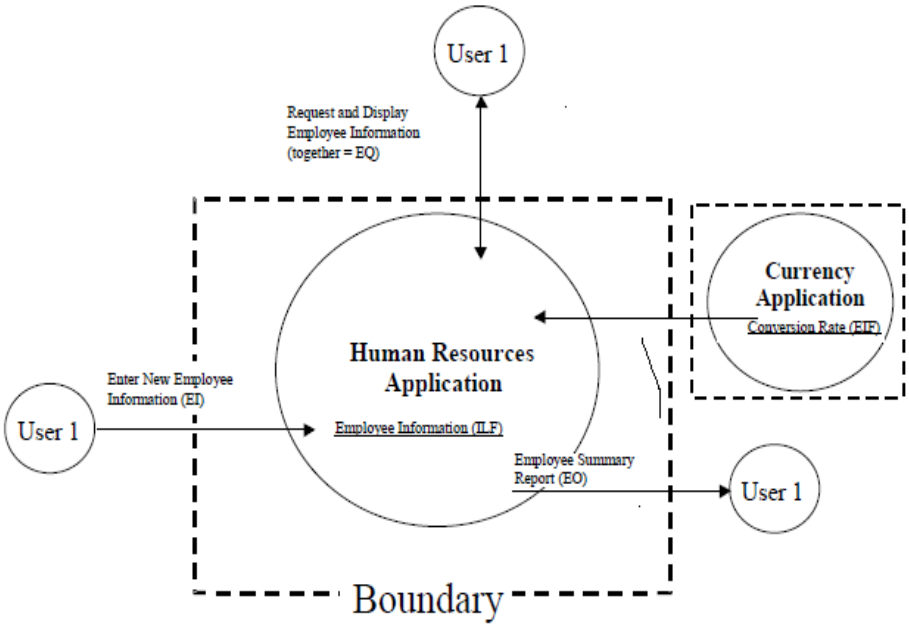


Figure 1-4 – Boundary example

Determine the Partition

Partition When identified, partitions may add non-functional size. Sub-category 1.4 (Internal Data Movements) is used to provide the additional non-functional size for the application being assessed.

Rules and Procedures

This section defines the rules and procedures that apply when determining assessment scope and boundary of the application(s).

Scope hints The following hints can help you to identify the assessment scope:

- Review the purpose of the non-functional assessment to help determine the assessment scope.
- When identifying the scope for the assessment of the non-functional size of the installed base of applications, include all of the non-functional categories supported by the maintenance team, eventually distinguished by partition within each application's boundary.

(cc) BY-NC-SA

Boundary Rules

The following rules shall apply for boundaries:

- The logical application boundaries need to be consistent between the FPA and SNAP processes.
- The boundary is determined based on the user view; the focus is on what the user can understand and describe.
- The initial boundary already established for the application, or applications being modified, is not influenced by the assessment scope.

Note:

There may be more than one application included in the assessment scope. If so, multiple application boundaries would be identified.

When the boundary is not well-defined (such as early in analysis), it should be located as accurately as possible.

Boundary and Partitions Hints

The following hints can help you to identify the boundary and the partition of the application(s):

- Use the system external specifications or a system flow chart and draw a boundary around it to highlight which parts are internal and which are external to the application.
- Look at how groups of data and software partitions are being maintained.
- Identify functional areas by assigning ownership of certain types of analysis objects (such as entities or elementary processes) to a functional area; non-functional categories are determined by the identification of the functional boundaries (Application boundaries as determined by FPA) and eventually partitions within them.
- Look at the associated measurement data, such as effort, cost, and defects; the boundaries for measurement should be the same, or eventually those measurement data might be distinguished by partitions within a single boundary.
- Interview subject matter experts for assistance in identifying the boundary.
- Interview software analysts for assistance in identifying the partitions, if any.

 (cc) BY-NC-SA

This page intentionally left blank

Categories & Sub-categories

Introduction This chapter presents the details behind categories & sub-categories including questions, ratings, and SCUs.

Contents This chapter includes the following:

Topic	Page
Categories and Sub-categories	5-2
Sub-category Complexity	5-2
Code Data	5-3
Definitions of additional terms used in this Manual	5-6
Category 1: Data Operations	5-9
1.1 Data Entry Validation	5-9
1.2 Logical and Mathematical Operations	5-12
1.3 Data Formatting	5-15
1.4 Internal Data Movements	5-17
1.5 Delivering Added Value to Users by Data Configuration	5-19
Category 2: Interface Design	5-21
2.1 User Interfaces	5-21
2.2 Help Methods	5-24
2.3 Multiple Input Methods	5-27
2.4 Multiple Output Methods	5-28
Category 3: Technical Environment	5-30
3.1 Multiple Platforms	5-30
3.2 Database Technology	5-34
3.3 Batch Processes	5-36
Category 4: Architecture	5-38
4.1 Component Based Software	5-38
4.2 Multiple Input / Output Interfaces	5-40
Mission Critical/Real Time Systems	5-42
SNAP Calculation Example	5-45

 (cc) BY-NC-SA

Categories and Sub-categories

Definition

See Chapter 2 Section 2 for the definition of categories and sub-categories

Sub-category Complexity

Complexity definition

Each sub-category is of a different nature; therefore, it may have a different set of parameters that define complexity. Complexity was defined by asking the following questions:

1. What are the main drivers that are considered by software project estimators as affecting the complexity of the item.

2. Assuming one small team has its one set of productivity values (skill set, methodologies, working environment etc.) – such a team will estimate that more work is needed to provide a complex item than to provide a medium item.

Complexity Parameters

The Parameters that are counted or evaluated in order to assess the Complexity.

In the example below (see table 1-3), the complexity parameters are:

- Number of Data Elements Types (DETs).
- Number of nesting levels.

Complexity Example

Complexity of Data Entry Validation may be defined by the number of nesting levels, and also by the amount of data elements that are used in the process. In this case, we may decide that parameter #1 will be based on the number of nesting levels, and parameter #2 will be the number of data elements.

Complexity grid

Use the following tables to illustrate SNAP Points calculation

	Nesting Level Complexity		
	Low	**Average**	**High**
	1-5	**6-14**	**15+**
SP=	2*#DETs	3*#DETs	4*#DETs

Table 1-3 Example 1 of SNAP Points calculation

(cc) BY-NC-SA

Another option to illustrate SNAP Points calculations is based on the type of the assessed item, for example:

Help Type	SP =
User Manual	1*(#help items)
On-line Text	2*(#help items)
Context Help	2*(#help items)
Context + On-line	3*(#help items)

Table 1-4 Example 2 of SNAP Points calculation

Code Data

Code data is a type of data entities, used for software sizing (in addition to Business Data and Reference Data).

According to IFPUG CPM (Part 3 Chapter 1), "Code Data" usually exists to satisfy non-functional user requirements from the user (for quality requirements, physical implementation and/or a technical reason).

The user does not always directly specify Code Data (Sometimes referred to as List Data or Translation Data). In other cases it is identified by the developer in response to one or more non-functional user requirements.

Code Data provides a list of valid values that a descriptive attribute may have. Typically the attributes of the Code Data are Code, Description and/or other 'standard' attributes describing the code; e.g., standard abbreviation, effective date, termination date, audit trail data, etc. The different categories of data are outlined below to assist in identification.

When codes are used in the Business Data, it is necessary to have a means of translating to convert the code into something more recognizable to the user. In order to satisfy non-functional user requirements, developers often create one or more tables containing the Code Data. Logically, the code and its related description have the same meaning. Without a description the code may not always be clearly understood.

The key differences between Code Data and Reference Data are:

- With Code Data, you can substitute one for the other without changing the meaning of the Business Data; e.g., Airport-Code versus Airport-Name, Color-Id versus Color-Description.
- With Reference Data, you cannot substitute (e.g., Tax Code with the Tax-Rate).

 (cc) BY-NC-SA

Code Data has most of the following characteristics:

Logical Logical Characteristics include:

- Data is mandatory to the functional area but optionally stored as a data file;
- Not usually identified as part of the functional user requirements; it is usually identified as part of design to meet non-functional user requirements;
- Sometimes user maintainable (usually by a user support person)
- Stores data to standardize and facilitate business activities and business transactions;
- Essentially static - only changes in response to changes in the way that the business operates;
- Business transactions access Code Data to improve ease of data entry, improve data consistency, ensure data integrity, etc.; and
- If recognized by the user:
    - is sometimes considered as a group of the same type of data.
    - could be maintained using the same processing logic.

Physical Physical characteristics include:

- Consists of key field and usually one or two attributes only;
- Typically has a stable number of records;
- Can represent 50% of all entities in Third Normal Form;
- Sometimes de-normalized and placed in one physical table with other Code Data; and
- May be implemented in different ways (e.g., via separate application, data dictionary, or hard-coded within the software).

Examples Examples of Code data include:

- State
    - State Code
    - State Name
- Payment Type
    - Payment Type Code
    - Payment Description

Handling Code Data from Non-functional Sizing Perspective

For the purpose of FPA, Code Data cannot be counted as logical files. They are not considered as Internal Logical File (ILF) or External Interface File (EIF), and cannot be considered Record Element Types (RETs) or Data Element Types (DETs) on an ILF or EIF. Code Data cannot be considered a File Types Referenced (FTR) while assessing the complexity of a transactional function (External Input -EI, External Output - EO, and External Inquiry - EQ).

For the purpose of SNAP, Code Data which is maintained within the application boundary by use of screens or by formal enhancement requests by the customer is counted as follows:

- Irrespective of the number of Code Data physical tables, the Code Data would be

 BY-NC-SA

grouped as 1 data group (1 FTR) under SNAP.

- Code data is classified as

Types of Code Data

Substitution	Static or Constant	Valid Values
Code + Description	One occurrence Static Data Default value	Valid Values Range of valid values

Table 1-5 Types of Code Data

- For SNAP analysis of complexity of Code Data group, the number of RETs of the code table would depend upon the type of occurrences of Code Data types.

Example

In a Banking application, the following Code tables were created:

1. Table 1: Having State name and State code
2. Table 2: Having Branch code, Branch name, Branch city
3. Table 3: Having one single data entry of the Bank Name and logo , which is used for printing letterheads

Three types of occurrences exist for Code Data

4. Substitution - Table 1 having State code and state name, Table 2 having branch code and branch name
5. Valid Values -Table 2 having range of bank branch cities
6. Static Data - Table 3 having one single data entry of the Bank Name and logo , which is used for printing letterheads

Hence the number of RETs for the Code data group is 3.

Note: If the data is not of the above code data sub types, then it may be the data in system tables and not the application tables required for supporting business. This is not sized under code data

How Code Data is sized using SNAP?

Count the creation / maintenance and the utilization of code data:

- The creation of Code Data is always counted as "3.2 Database Technology."
- The maintenance of Code Data is checked under the following subcategories depending on the case applicable
    - If the Code Data is maintained in hard coded tables which are not viewable via screens but can be updated by the administrator using script/code change in response to a formal user request, then it is counted using sub-category 3.2: Database technology.
      For example, a code table has the bank name and logo stored as static data, which is referred by different processes. When a change request is raised to modify the logo, then it is sized using this category.
    - When the Code Data is used for reasons such as entry validations, data formatting, batch process management, any changes to code data values (Add /Change/ Delete) will be counted using the proper sub-category.

 (cc) BY-NC-SA

- The utilization of Code Data is counted in the following sub-categories, according to the purpose of the data: "1.1 Data Entry Validation"; "1.2 Logical and Mathematical Operations"; "1.3 Data Formatting"; "1.5 Delivering Added Value to Users by Data Configuration" and "3.3 Batch Processes."
- When non-functional requirements use Code Data and transactions cross partitions, sub-category "1.4 Internal Data Movements" should be used.

Examples of SNAP sizing of Code Data

Examples:	Sub-Categories for Utilizing Code Data
1. Create a Code Data table for address validation	1.1 Data Entry Validation for enabling the validation 3.2 Database Technology for code table creation
2. Same as above. The screens with the address are on Front-End application, the data is in the Back-End application	1.1 Data Entry Validation for enabling the validation 3.2 Database Technology for code table creation 1.4 Internal Data Movements
3. Using multi-language screens, the translation is in new Code Data tables	1.3 Data Formatting 3.2 Database Technology for code table creation
4. Add scheduling data to perform batch files	3.3 Batch Processes 3.2 Database Technology for code table creation

Table 1-6 Example of SNAP sizing of Code Data

Refer to the subcategories and to the examples in Part 1, Chapter 5 below

Definitions of Additional Terms used in this Manual

Elementary Process (EP)
An elementary process is the smallest unit of activity that is meaningful to the user(s). The elementary process must be self-contained and leave the business of the application being counted in a consistent state

Data Element Type (DET)
A DET (Data Element Type) in this manual, is a unique, non-repeated attribute, which can be in Business Data, Reference Data, or Code Data.
Count the number of different types of Data Elements of all tables as the number of DETs.

# of DETs
The sum of all DETs which are part of the input + output of the elementary process, plus the data elements which are read or updated internal to the boundary.

 BY-NC-SA

Record Element Type (RET)

User recognizable sub-group of data element types within a data function, and Code Data group as defined in the "Code Data" paragraph.

Logical File

A logical file is a logical group of data as seen by the user. A logical file is made up of one or more data entities.

A data function represents functionality provided to the user to meet internal and external data storage requirements. A data function is either an internal logical file or an external interface file.

Grouping of data into logical files is the result of the combined effect of two grouping methods:

- Method a) is process driven, based on the user transactions in the application.
- Method b) is data driven, based on the business rules.

File Type Referenced (FTR)

A file type referenced is a data function read and/or maintained by a transactional function. A file type referenced includes:

- An internal logical file read or maintained by a transactional function, or
- An external interface file read by a transactional function
- Code Data is grouped to one additional FTR

Database View

In database theory, a database view is the result set of a stored query on the data, which the database users can query just as they would in a persistent database collection object. This pre-established query command is kept in the database dictionary. Unlike ordinary base tables in a relational database, it is a virtual table computed or collated dynamically from data in the database, when access to that view is requested. Changes applied to the data in a relevant underlying table are reflected in the data shown in subsequent invocations of the view.

Constant Factor

A multiplier used to calculate the number of SNAP points. The SNAP point size is a result of the [constant factor] times [a complexity parameter].

Example:

SP =	2*#DETs

2 is the constant factor

#of DETs is the complexity parameter

 BY-NC-SA

Single / Multiple Instance approach

Different organizations may take different approaches for sizing similar functionality being delivered on different medias. They use the single instance or the multiple instance approaches to specify the same. Single instance approach is said to be when the same functionality is delivered via different mediums (input or output), but is counted only once.

Multiple instance approach is the case where each method of delivery of same functionality is counted separately.

Organizations using the single instance approach for the FP size can size the other methods of delivery using SNAP.

Single Instance approach

The single instance approach does not recognize the medium for delivery for a transaction function as a differentiating characteristic in the identification of unique transaction functions. If two functions deliver the same functionality using different media, they are considered to be the same function for functional sizing purposes.

Multiple Instance approach

The multiple instance approach specifies that instance functional size is taken in context of the approach objective of the count, allowing a business function to be recognized in the context of the medium in which it is required to operate.

The multiple instance approach recognizes the medium for delivery for a transaction function as a differentiating characteristic in the identification of unique transaction functions.

 BY-NC-SA

Category 1: Data Operations

Data Operations

The Data Operations category relates to how data is processed within the SCU to meet the non-functional requirements in the application.

1.1 Data Entry Validation

Definition

Operations that are performed either to allow only certified (predefined) data or to prevent the acceptance of uncertified data

SCU

The elementary process.

Terms

Nesting Level:

The number of conditional validations (If-Else combo/"While" loop/"For" loop or any other validation blocks) in the longest chain of validation

Complexity Parameters:

1. Nesting level complexity
   a. Low complexity: 2 nesting levels or less.
   b. Average complexity: 3 to 5 nesting levels.
   c. High complexity: 6 nesting levels or more.
2. Number of unique DETs used across all validations.
   For example, there are two fields validated in the SCU, one uses three DETs for nested validation and another uses one DET which is not one of the 3 DETs above, count Four DETs.

SNAP Points calculation

Identify the complexity based on nesting level. Calculate SNAP Points based on the constant factor and the number of DETs (#DETs).

	Nesting Level Complexity		
	Low	**Average**	**High**
	1-2	**3-5**	**6+**
SP =	2*#DETs	3*#DETs	4*#DETs

Table 1-7 SNAP sizing for Data Entry Validations

 BY-NC-SA

Notes:

- Data entry may result from any source (UI, transaction).
- Number of nesting levels is derived from the business requirements and high level solution and not from how the code is written.
- Validations are nested when there is a dependency between validations.
  Example: A number must be between 0 and 10: if it is less than 1, it must have two digits after the decimal point. If it is 1 or more, it may have one or no digits after the decimal point.
- Several validation on a field are not nested when they are independent.
  Example: a value must be numerical, greater than 0 and smaller than 1,000,000.
- This sub-category may include requirements for error handling or exceptions handling.
- DETs refer to all types of data. See Part 1 Chapter 5 (Definitions of Additional Terms) for the definition of data elements.
- If Code Data is used for data entry validations, then any changes to Code Data values (Add/Change/Delete) will be counted using this category.

Examples

Examples of Data Entry Validations: A date field must have a certain size; a value entered must be in a certain range of values; a code must be present in a certain table; a field is bound to the value of the previous field (e.g. State, County, City)

1: Data Entry validation enabled using code data for validation of Airport names

A travelling order application has a screen with details of the departure airport, destination airport and the option to add multiple destinations.

The current system validates the airport abbreviations (such as LHR, NYC) but cannot identify airport name.

The requirement is that the user will be able to key in either the abbreviation or the airport name.

The design is to use an existing Code Data with all airports and IATA Airport Codes, and to add validation rules both for the airport abbreviation and airport name.

Three elementary processes were identified (order a flight, amend order, cancel order) using this code data validation. One nesting level and one DET are used for SNAP counting.

 BY-NC-SA

2: Data Entry validation enabled using logical checks on the DETS - adding an employee to an organization

In the processing logic of adding an employee to an organization, the number of validations performed during data entry on a screen and the complexity level of these validations, are considered non-functional.

Employee ID, in this example, is not generated automatically but manually entered besides employee name, department, date of birth and more. Validating that the set of data is correct is sized using this subcategory.

(These are considered to be technical requirements).

As per the CPM, some operations, which are performed in the elementary process to ensure valid data, are added into the data information. The data elements in these operations are not visible to the user (although agreed with the user) and not counted as DETs in FP. Data elements that are involved in these hidden operations should be counted using SNAP model.

See also Part 2 Chapter 1 example 1

 (cc) BY-NC-SA

1.2 Logical and Mathematical Operations

Definition

Extensive logical decisions, Boolean operations, and extensive mathematical operations applied on the process.

Extensive Mathematical Operations

SNAP defines an "extensive mathematical operation" as a mathematical operation which includes using one or more algorithms. An algorithm is defined for SNAP as a series of mathematical equations and calculations executed in conjunction with, or according to, logical operators to produce results identifiable to the user. Examples of extensive mathematical operations include using the Program Evaluation Review Technique (PERT) to calculate the expected completion date of a project, calculating the optimal profit for a business process using linear programming, determining the way to formulate the fastest flowing waiting lines using queuing theory, and finding the shortest route through a network. Examples of other algorithmic mathematical operations fitting the definition of "extensive" include solving calculus integration formulas, calculating federal income taxes, GPS calculations, gaming, weather forecasting, and perhaps calculating retirement pensions.

The DETs counted are the set of those required to operate the extensive mathematical operation, such as values for an algorithm's variables and settings maintained by the algorithm's control information. These values and settings are not necessarily stored in a single physical file; they may be stored in various locations such as settings of the value of variables located in the code or as DETs in various physical files. However located, as a set(s) this satisfies the requirements for either an internal logical file(s) or external interface file(s) because they are the logical grouping(s) of data necessary to operate the algorithm.

"Simple" or "routine" mathematical operations are defined here as not using algorithms. Examples can include adding a column of numbers, balancing a checking account, totaling daily sales, and calculating averages. Also, an application may require a simple or routine mathematical operation to be iterated many times. For example, a fast food restaurant manager may need to place an order for ketchup packets from a supplier. The manager first counts the current inventory of ketchup packets, forecasts the expected usage, and places an order to make up for the expected resulting shortfall. This is a simple or routine mathematical operation. If the manager has 100 types of items in inventory, and must perform this calculation 100 times to complete the total order with the supplier, then this is still defined as being a simple or routine mathematical operation because the simple or routine mathematical operation is iterated 100 times: "extensive" refers to the depth of the algorithm(s), not to the number of simple or routine calculation iterations needed."

Sizing an elementary process is determined by the type of processing logic used by the external input (EI), EO, or external inquiry (EQ). While this can give a higher size to the elementary process that contains mathematical operations, it does not necessarily correlate to the effort needed to produce

 BY-NC-SA

extensive mathematical operations. SNAP size compensates for the additional complexity of extensive mathematical operations.

Extensive Logical Operations

SNAP defines an "extensive logical operation" as a logical operation either containing a minimum of 4 nesting levels, containing more than 38 DETs required to operate the logical operation, or both. These DETs do not necessarily have to cross the application boundary.

The outcome of a logical operation may be a decision or set of decisions or evaluating a condition using data that exist in one or more logical files. The SCU is the elementary process. If more than one logical operation can be executed within the elementary process, then count either the combined number of DETS in the operations containing a minimum or four nesting levels, or count the sum of the DETs involved in all of the logical operations assuming that the sum is more than 38 (whichever is larger).

SCU

The elementary process.

Complexity Parameters:

1) FTR density of the logical file being accessed to do the business logic processing - table 1-8.
2) Processing logic type of the elementary process (logical / mathematical) – table 1-9.
3) Number of data elements types (#DETs) – table 1-10.
   a) FTR density Factor is measured as follows

	FTR density		
	0-3 FTR	**4-9 FTR**	**10+ FTR**
	Low	Average	High

Table 1-8 FTR Density, Logical and Mathematical Operations

   b) Type of the elementary process (logical / mathematical)

   Identify the type of the elementary process

 (cc) BY-NC-SA

EP Type	Main Purpose of the EP
Logical	Decision making or evaluating a condition using data that exist in one or more logical files (internal and / or external) Example: Exception processing
Mathematical	Transformation of data and / or use of control information that exist in one or more logical files (internal and / or external) that is used for an extensive mathematical operation. Example: Complex tax calculation

Table 1-9 EP type for Logical and Mathematical Operations

Note:

- When the main purpose cannot be clearly identified, select "Logical" (Do not count it as one Logical and one Mathematical).

SP calculation

Calculate size based on the constant factor and the FTR density factor.

	Complexity Level		
	Low	**Average**	**High**
	SP=	**SP=**	**SP=**
EP type: Logical	4*#DETs	6*#DETs	10*#DETs
EP type: Mathematical	3*#DETs	4*#DETs	7*#DETs

Table 1-10 SNAP sizing for Logical and Mathematical Operations

Examples

- Project scheduling critical path analysis.
- Complex tax calculations.
- Linear programming algorithms.
- Calculus integration formulas.
- Financial return on investment calculations for a large industrial machine.
- Statistical Analysis of Variance calculations.
- Business sales forecasting using the ensemble forecasting method.

 (cc) BY-NC-SA

1.3 Data Formatting

Definition A requirement that deals with structure, format, or administrative information in a transaction not directly relevant to functionality that is seen by the user.

SCU The elementary process.

Complexity Parameters:

1. Transformation complexity:
    a. Low: Data type conversions or simple formatting such as byte padding, or data substitution, using a maximum of 2 operators (Celsius to Fahrenheit, Single Integer to Double Integer)
    b. Average: Involves encryption / decryption which is a characteristic of the application and applies to almost all processes, which is provided through a library – API interface
    c. High: Involves local Encryption/Decryption.
2. Number of data elements types (#DETs) transformed

Notes:

- Data elements refer to all types of data. See Part 1 Chapter 5 (Definitions of additional terms used in this chapter) for the definition of data elements
- Encryption algorithm is complex for:
    - Design specifically allowed to several key lengths;
    - Provide a method to ensure data integrity for high volume data;
    - Formatting medical images;
    - Restructure huge volume database etc.

SNAP Points calculation Identify the complexity based on transformation. Calculate SNAP Points based on the constant factor and the number of DETs (#DETs).

	Transformation Complexity		
	Low	**Average**	**High**
SP =	2*#DETs	3*#DETs	5*#DETs

Table 1-11 SNAP sizing for Data Formatting

Notes: It may include adherence to standards, layouts, messages.

 (cc) BY-NC-SA

Examples (See note below)

1. Simple transformations:
   - Convert text to value or value to text; convert strings to values.
   - Data formatting required for reporting requirements.
   - An application shows the date (MMDDYYYY), time (GMT), current atmospheric temperature (degrees Fahrenheit) in a standard format. However, due to regulations, the date is required to be displayed as 'YYYYMMDD, the time should always show the local time zone, and temperature should be displayed as "Degrees Kelvin."
     As a result the display formats needs to be converted to adhere to the standards prescribed.
2. Complex transformations:
   - Enabling multi-lingual support for an application by using Code Data
   - Encryption/ Decryption, Compression - Decompression
   - Compliance to standards for electronic transfer of data in healthcare environment. The data packets are sent and received in particular format of EDI transactions.
     For example: Change the structure of the transactions - add headers and footers; the transaction format is changed per HIPAA (Health Insurance Portability and Accountability Act of 1996) requirements with no changes in the functionality.
   - Data interchange formats - XML to other formats, or other means of data interchange between two computer systems.
   - Preparation of metadata for various screen requirements or data warehouse views.
   - Transformations in data warehouse.

Note: When a transformation is agreed as functional between the user and the development team, and transformation is FP counted, do not add SNAP Points. If it is agreed as NFR between user and development team, use SNAP

See also Part 2, Chapter 1, example 2.

1.4 Internal Data Movements

Definition

Data movements from one partition to another within application boundary with specific data handling..

SCU

The portion of the elementary process, which crosses from one partition to another.

The SCU is identified by the elementary process and the two partitions crossed.

Notes

1. Elementary process, as defined by the CPM (Part 5, Glossary), is the smallest unit of activity that is meaningful to the user. While the CPM refers to transactions that cross the application boundary - for sub-category 1.4 SCU - the APM refers to their internal processes/functions, which move from one partition to another.
2. If an elementary process crosses more than two partitions, use the formula below per each partition crossing (in figure 1-3, an elementary process may move from component 1 to component 2 (Labeled "A"), and then to component 3 (Labeled "B"). In such case, SNAP Points will be calculated at each partition crossing).

Complexity Parameters:

1. Number of unique data elements types (#DETs) transferred from one partition to the other, and are processed and / or maintained.
2. Number of unique FTRs either read or updated by the elementary process at both partitions crossed.

SNAP Points calculation

Identify the complexity level based on the number of FTRs read/updated and number of DETs transferred. Calculate size as per the table below for each partition crossing:

	Complexity Level		
	Low (0-3 FTR)	**Average (4-9 FTR)**	**High (10+ FTR)**
SP =	4*#DETs	6*#DETs	10*#DETs

Table 1-12 SNAP sizing for Internal Data Movements

Notes:

1. Internal Data Movements sub-category sizes internal transactions within the boundary of an application. These transactions are used in case they cross partitions.
2. Internal data movements are counted by SNAP for functional transactions as well as non-functional transactions.
   For example: Querying fields on a front-end application by using data that is stored in the back-end application

 BY-NC-SA

3. Any data transaction (functional) that crosses partitions, generates SNAP points in addition to Function Points.
4. When an elementary process crosses the partition in both directions, count SNAP as follows:
   a. One SCU, if the transactions are synchronous
   b. Two separate SCUs, if the transactions are a-synchronous

- The following are examples of elementary processes that may have data crossing partitions Data backup within application boundary, crossing partitions.
- Data copy/movement between tables within application boundary, crossing partitions.
- Realign data in temporary storage.
- Transactions between application and middleware that do not cross functional boundaries.
- SOA (Service Oriented Architecture) solutions. (When SOA functionality are within the application boundary).
- Data movements between tiers that do not cross functional boundaries.
- Data formatting transactions which use data that crosses partitions.
- Internal transactions for logical /mathematical operations.
- Reference Data Maintenance.

Sub-Category 3.3 (Batch Processes) covers batch jobs. Batch jobs may be executed within a single partition. If the batch job crosses partitions, then it may need to be sized for additional impacts by this sub-category

Example

An elementary process "process invoice" has two partition crossings as shown in figure 1-3a.

Each component references/updates a unique set of FTRs during the processing as follows:

Partition 1 - 2 FTRs

Partition 2 - 4 FTRs

Partition 3 - 3 FTRs

For this process six (6) DETs are crossing from partition 1 to partition 2 and five (5) DETs are crossing from partition 2 to partition 3.

For the SCU "A crossing":

Number of FTRs = 2+4 =6 (Average complexity)

Number of DETs = 6

SP = 6*#DETs = 36

For the SCU "B crossing":

 BY-NC-SA

Number of FTRs = 4+3 =7 (Average complexity)

Number of DETs = 5

SP = 6*#DETs = 30

1.5 Delivering Added Value to Users by Data Configuration

Definition

Additional unique business value to users that is provided by adding, changing or deleting reference data/ code data information from the database or data storage with no change in software code or the database structure.

SCU

The elementary process per logical file

Notes: The SCU is the elementary process to consume the added value in the logical file and not the process to create or modify the configuration

Example: A new service is defined by adding its attributes to reference tables. The application is flexible enough to provide the new service with no code changes. Elementary processes to be counted may be: add the new service; change this service; cease the service. The process to add the service attributes to the reference tables (i.e., writing and using scripts) should not be counted.

In case the configured data impacts several elementary processes, each elementary process is counted separately

Terms

Attribute

An independent parameter that has a unique business meaning and contains a set of different values

A record

One row in a logical file

A Logical File

A user recognizable group of logically related data or control information.

Complexity Parameters:

1. Number of unique attributes involved in the elementary process, that are added / modified / deleted
2. Number of Records configured

SNAP Points calculation

Identify the complexity level based on #Records. Calculate size based on the constant factor and the #Attributes.

	Complexity Level		
	Low	**Average**	**High**
	1-10 records	**11-29 records**	**30+ records**
SP =	6*#attributes	8*#attributes	12*#attributes

Table 1-13 SNAP sizing for Data Configuration

Notes:

New services, products, price plans etc., can be added to the application by adding or changing reference data and not by writing code.

Functionality by Data Configuration brings added value to the user, also adds effort to configure and test the new functionality.

Examples

This sub-category sizes functionality that is created by adding data to the database.

1. Application requires granting access to specific role in the application. To meet this requirement, developer does not write any separate code and instead updates a configuration file, and associates the user or set of users into some property file(s). Such additions or changes are made to meet user requirements, which affect the functionality at the elementary process level.

The process to configure the data into the database is not sized separately. Only the user's processes should be counted.

2. Application requires configuring a new product ("Product X here below) or a component that can be sold using the application. The new product and its price plan are defined in reference data. The project effort may be creating the data, by migrating it to the reference files and testing that the application functions with the new data. The assessment identifies many SCUs here.

- Change product Y to product X.
- Provide product X.
- Change price of product X etc.

See also Part 2, Chapter 1, example 4.

Category 2: Interface Design

Interface Design

The Interface Design Category relates to the end user experience. This category assesses the design of UI processes and methods that allow the user to interface with the application.

2.1 User Interfaces

Definition

Unique, user identifiable, independent graphical user interface elements added or configured on the user interface that do not change the functionality of the system but affect non-functional characteristics (such as usability, ease of learning, attractiveness, accessibility)

SCU

Set of screens as defined by the elementary process

Terms

UI Element

UI Element (User Interface Element) is a unique user identifiable structural element which makes up the user interface. It includes elements such as:

1) Window (which can be container, child, text or message window)
2) Menus
3) Icons
4) Controls
    a. Pointer (or mouse cursor)
    b. Text box
    c. Button
    d. Hyperlink
    e. Drop-down list
    f. List box
    g. Combo box
    h. Check box
    i. Radio button
    j. Cycle button
    k. Datagrid
5) Tabs
6) Interaction elements like a cursor
7) Labels

The above controls are used to display or manipulate data objects. The aspect that adds to complexity of the design, configuration and testing time of a user interface is the configuration of each of these elements. Non-functional requirements may involve changing the properties of these UI Elements. Depending upon the type of the user-interface element, varying number of properties can be configured to produce a desired output. For Example, Button could be set to “locked” or “highlighted” or “colored” or placed at a particular location on the screen.

 (cc) BY-NC-SA

UI Element Properties

Each UI Element is associated with certain properties which define the behavior and look and feel of the User Interface Element. For example, a window would have properties like: background color, active border, active caption etc.

A button can have properties like: ButtonHighlight, ButtonText, BackgroundColor etc.

Tool Tips can have properties like: Info Text, Info Background etc.

The above examples have been cited from W3C recommendations. [2]

UI Element Set

A UI element set is the collection of all the UI elements of the same type in the SCU.

Example: All the text boxes in the set of screens (SCU).

Complexity Parameters:

1. The sum of the number of unique properties configured for each UI element in the SCU.
2. Number of unique UI elements impacted.

SNAP Points calculation

Identify the complexity based on number of properties of UI element set. Calculate size as the product of the constant factor and the number of unique UI elements.

	UI Type Complexity		
	Low	**Average**	**High**
	<10 properties added or configured	**10 - 15 Properties added or configured**	**16+ Properties added or configured.**
SP =	2*# unique UI elements	3 * # unique UI elements	4*#unique UI elements

Table 1-14 SNAP sizing for User Interfaces

Rules

1. If the process for adding/changing of UI is FP counted, then do not duplicate the count in the assessment; however, changing the contents and appearance of a GUI element needs to be assessed as non-functional. Aesthetic changes in UI screen, Static or Dynamic UI Pages, Rearranging of screen and printed reports should be assessed under User Interfaces sub-category.
2. The sets of screens within one process will be counted as the SCU

[2] www.w3.org/TR/CSS2/ui.html

 BY-NC-SA

Notes:

UI Elements added may be of any form, such as text, sound, picture, logos, color, keys, controls, navigation, animating or enabling/disabling the above (when such enabling/disabling is not functional)

Added operations that support: Function Keys; Auto fill; Short cut keys; Common Keys; Navigation - Screen/Page level

Screens that are not considered functional (such as Administrator's screens) and hence are not counted using Function Points, are counted by SNAP using 2.1 User Interfaces sub-category.

Example

Some text on users' screens is hard coded. Due to a new policy of the company, the request is to replace the word "customer" with the words "Business partner."

The analysis found that the word "customer" should be replaced in the following UI elements (Note: since SNAP counts the number of unique UI elements, there is no need to estimate the number of occurrences of each unique UI element)

SCU 1: Acquire a new business partner:

Header, labels, radio button, drop-down list

SCU 2: Modify details of a business partner:

Header, labels

SCU 3: Send message to a business partner:

Header, labels,

SCU 4: Cease services to a business partner:

Header, labels

Changing the text in these UI elements is considered one property. UI type complexity is Low

SP = 2*# unique UI elements per each SCU:

SP = 2*(4 + 2 + 2 + 2) = 20

See also Part 2, Chapter 1, example 5.

 (cc) BY-NC-SA

2.2 Help Methods

Definition Information provided to the users that explains how the software provides its functionality or other supportive information provided to users.

SCU The assessed application.

Terms

Help item

A Help Item is the smallest, unique, user identifiable help or information topic which provides the user supportive information or details about a particular part of software.

Context Help

Context Help refers to a help program feature that changes depending on what user is doing in the program. It is a kind of online help that is obtained from a specific point in the state of the software, providing help for the situation that is associated with that state.

Context-sensitive help can be implemented using tooltips, which either provide a brief description of a GUI widget or display a complete topic from the help file. Other commonly used ways to access context-sensitive help start by clicking a button. One way uses a per widget button that displays the help immediately. Another way changes the pointer shape to a question mark, and then, after the user clicks a widget, the help appears.

Static Web Page

A static web page is a web page that is delivered to all the users exactly as stored, displaying the same information to all users, and is not generated by an application.

Complexity Parameters:

1. Help Type
   a. User Manual (Hard copy/ soft copy / Application level help)
   b. On-line Text
   c. Context
   d. Context + On-line
2. Number of help items impacted

SNAP Points calculation Identify the help type. Calculate size based on the constant factor and the number of help items impacted.

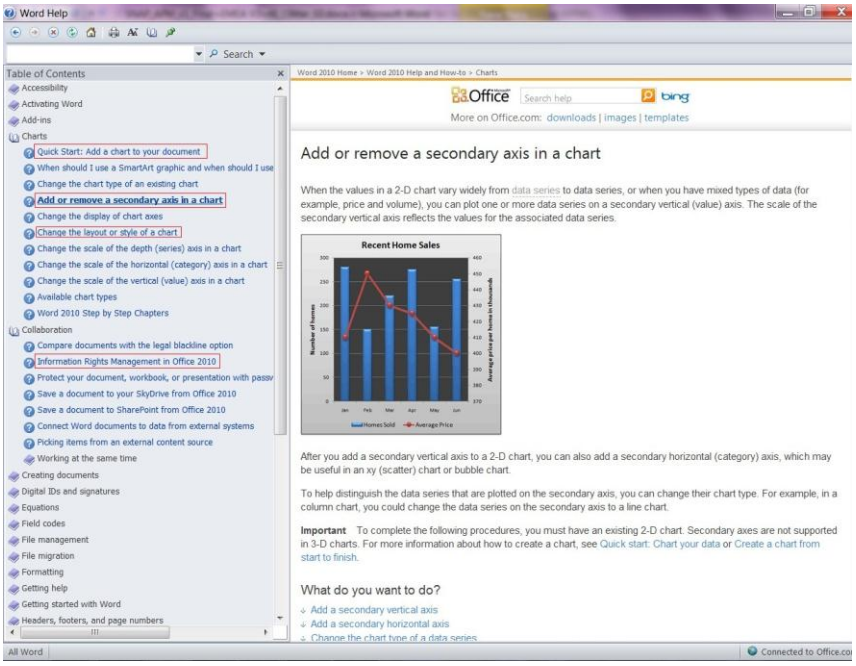
 BY-NC-SA

Help Type	SP =
User Manual	1*(#help items)
On-line Text	2*(#help items)
Context Help	2*(#help items)
Context + On-line	3*(#help items)

Table 1-15 SNAP sizing for Help Methods

Example

A good example of Help Item can be given from Windows Help. When we click F1 in MS Word/Excel, a help window appears and we can see a Table of Contents in the left. By clicking each of the Help Content items, we can see the granular level of help which has the details of the help sub-topic. Each of these smallest granular help topics can be an individual Help Item. A screenshot showing the same is given below. The Help Items are encircled in Red.



See also Part 2, Chapter 1, example 6.

 BY-NC-SA

Notes:

Static web pages

Use the above explanation to size static web pages. Although a static page is not directly a "help item", the Help methods subcategory should be used to identify the complexity' and then calculate the non-functional size.

Although there may be UI aspects in building and maintaining static web pages, do not add size using sub-category 2.1. The primary intent of 2.1 is to address GUI changes to improve usability, look and feel, learnability etc. of the functionality of the application

Help items that involve User Interfaces

There are many cases in which adding a Help item involves UI effort. In such cases, it is not expected to size this activity twice, as SNAP Points for Help sub-category and additional SNAP Points for User Interfaces.

When the primary intent of the activity is creating a Help item, only this sub-category should be used

The calibration of the equations in this sub-category remains open for future research based on pilot data from industry.

 (cc) BY-NC-SA

2.3 Multiple Input Methods

Definition

The ability of the application to provide its functionality while accepting multiple input methods.

SCU

The elementary process.

Terms

Input Method

A technique or media type, which is used to deliver data into the assessed application, such as bar code reader, fax, PDF, office document, screen, voice message, SMS, smart mobile device etc.

The assessed application may need to identify the input method in order to interpret and use the received information.

Complexity Parameters:

1. The number of data element types (DET's) in the SCU.
2. The number of additional input methods.

SNAP Points calculation

Identify the complexity based on the number of DET's. Calculate size based on the constant factor and the number input methods.

Note:

When counting a new development project, the number of input methods should include the additional ones only, assuming that one of the input methods is the base method.

For example, if the new development uses 4 input methods, the number of additional input methods is 3.

	Input Methods Complexity		
	Low	**Average**	**High**
	1-4 DET's	**5-15 DET's**	**16+ DET's**
SP=	3*# additional input methods	4*# additional input methods	6*# additional input methods

Table 1-16 SNAP sizing for Multiple Input Methods

Rules

This category should be used to when there are multiple types of inputs used to invoke the same functionality. If the different input types differ in terms of DETs, FTRs and processing logic, then they would already have been accounted as separate functions in function point counting process.

If they are same, then multiple input methods should be used.

Check the following

1) Approach taken for FP counting - single instance or multiple instance
2) The multiple methods of input for the same functionality (Same DETs, FTRS and processing logic) have not been included for FP size calculation. In other words, if the FP count has been done using single instance approach for different media types, then the additional input

method of same data entry needs to be accounted for using SNAP. For example the same input can be provided via a smart phone or a web screen.

3) If multiple input methods are already accounted for in the FP count or the multiple instance approach has been taken for FP counting, then it should be excluded from the SNAP assessment.

Example See also Part 2 Chapter 1 example 7

2.4 Multiple Output Methods

Definition The ability of the application to provide its functionality while using multiple output methods.

SCU The elementary process

Terms

Output Method

A technique or media type, which is used to deliver data from the assessed application, such as fax, PDF, office document, screen, voice message, SMS etc.

The assessed application may need to manipulate the sent information in order to send it to the various outputs.

Complexity Parameters:

1. The number of DATA Element Types (DET's) in the SCU
2. The number of additional output methods

SNAP Points calculation Identify the complexity based on the number of DET's. Calculate size based on the constant factor and the number output methods.

Note: When counting a new development project, the number of output methods should include the additional ones only, assuming one of the output methods is the base method.

For example, if the new development uses 4 output methods, the number of additional output methods is 3

	Output Methods complexity		
	Low	**Average**	**High**
	1-5 DET's	**6-19 DET's**	**20+ DET's**
SP =	3*# additional output methods	4*# additional output methods	6*# additional output methods

Table 1-17 SNAP sizing for Multiple Output Methods

Rules This category should be used to when there are multiple types of outputs used for the same functionality. If the different output types vary in terms of

BY-NC-SA

DETs, FTRs and processing logic, then they would already have been counted as separate functions in function point counting process.

If they are same, then multiple output methods should be used.

Check the following

1) Approach taken for FP counting – single instance or multiple instance
2) The multiple methods of output for the same functionality (same DETs, FTRS and processing logic) have not been included for FP size calculation. In other words, if the FP count has been done using single instance approach for different media types, then the additional output method of same data entry needs to be accounted for using SNAP.

   For example the same output can be provided to a smart phone or to a web screen.
3) If multiple output methods are already accounted for in the FP count or the multiple instance approach has been taken for FP counting, then it should be excluded from the SNAP assessment.

Example See also Part 2, Chapter 1, example 7.

 BY-NC-SA

Category 3: Technical Environment

Technical Environment

The Technical Environment category relates to aspects of the environment where the application resides. It assesses technology as well as changes to internal data and configuration that do not provide added or changed functionality from a Function Points perspective.

3.1 Multiple Platforms

Definition

Operations that are provided to support the ability of the software to work on more than one platform.

Note:

In order for software to be considered multi-platform, it must be able to function on more than one computer architecture or operating system. This can be a time-consuming task given that different operating systems have different application programming interfaces or APIs (for example, Linux uses a different API for application software than Windows does).

SCU

The elementary process

Terms:

Computing platform includes a hardware architecture and a software framework (including application frameworks), where the combination allows software, particularly applications software, to run. Typical platforms include a computer's architecture, operating system, programming languages and related user interface (run-time system libraries or graphical user interface).

Software Platforms: Software Platform is a framework used for the software application development. Different programming languages can be grouped into several platforms based on the programming language family.

A programming language can belong to a particular software language family like Object Oriented, Procedural, Declarative etc.

Object Oriented: Java, C++, C#, Javascript, Phython, Smaltalk, VB, VB.NET etc

Procedural: C, FORTRAN, PHP, COBOL etc.

Declarative: SQL, XQuery, BPEL, XSLT, XML etc.

Hardware Platforms: A hardware platform can refer to a computer's architecture or processor architecture. For example, the x86 and x86-64 CPUs make up one of the most common computer architectures in use in general-purpose home computers.

Complexity Parameters:

1. Nature of platforms (i.e., software, hardware)
2. Numbers of platforms to operate

 BY-NC-SA

SNAP Points calculation

Identify the different software and hardware platforms involved and the number of platforms to operate. Calculate size based on the platform category row from the table below and the number of platforms. If more than one row is applicable, then the size is sum of constant factors obtained from each category applicable.

	SP =		
	2 platforms	3 platforms	4+ platforms
Category 1 : Software Platforms: Same Software Family	20	30	40
Category 2: Software Platforms: Different Family	40	60	80
Category 3: Software Platforms: Different Browsers	10	20	30
Category 4: Hardware Platforms: Real time embedded systems	TBD*	TBD*	TBD*
Category 5: Hardware Platforms: Non-Real time embedded systems	TBD*	TBD*	TBD*
Category 6: Combination of Hardware and Software: Non-Real time embedded systems	TBD*	TBD*	TBD*

Table 1-18 SNAP sizing for Multiple Platforms

*TBD: To be Defined

Notes:

1. Working on one platform is a basic requisite for the system to operate; therefore a single platform does not generate SNAP points using this sub-category.
2. Building the software on multiple platforms generates SNAP points for two platforms or more according to table 1-18.
3. Software platforms can be added or removed, not changed. For example, upgrading from Firefox 3.1.x to 3.2.a is considered as adding a platform, not changing one.
4. Upgrading a software platform from one version to another is counted as adding a platform only. The old platform is not counted as deleted nor as changed.
5. When adding or removing platforms during an enhancement project, size the Multiple Platform sub-category for each impacted SCU at the end of

 BY-NC-SA

the enhancement project and use that as the SNAP size for the enhancement.

6. For enhancement projects, count the total number of platforms after the project is complete and use that to calculate SNAP size (Do not size the changes, additions or deletions of platforms during the enhancement project).
7. When adding a mixture of platforms from different categories, count SNAP Points for each platforms category.
   a) Example 1
   2 platforms of Family 1 and 2 platforms of Family 2:
   Use category 1 (similar platforms) for each software family (20SP for the two platforms of family 1)+ (20SP for the two platforms of family 2)
   Use category 2 (different platforms) - SNAP points for 2 different platform families (=40 SP)
   b) Example 2
   3 platforms of Family 1 and 2 platforms of Family 2:
   Use category 1 (similar platforms) for each software family (30SP for the 3 platforms of family 1)+ (20SP for the two platforms of family 2)
   Use category 2 (different platforms) - SNAP points for 2 different platform families (=40 SP)
   c) Example 3
   3 platforms of Family 1 and 1 platform of Family 2:
   Use category 1 (similar platforms) for each applicable software family (30SP for the 3 platforms of family 1)+ (0 SP for the 1 platform of family 2)
   Use category 2 (different platforms) - SNAP points for 2 different platform families (=40 SP)
   d) Example 4
   3 platforms of Family 1, 1 platform of Family 2 and 2 browsers:
   Use category 1 (similar platforms) for each software family (30SP for the 3 platforms of family 1) + (0 SP for the platform of family 2)
   Use category 2 (different platforms) - SNAP points for 2 different platform families (=40 SP)
   Use Category 3 for the browsers support (10 SP)

Calculation example

If an application is built on JAVA and COBOL and requires multiple (more than 4) browser support, then SNAP size would be

40 SP per each SCU that is built on both Java and Cobol, (java and COBOL are considered as different family

plus

30 SP per each SCU that is to work on multiple browsers

 BY-NC-SA

Example

- Software Platform: .NET, Java
- Operating System Platform: MS Windows, Linux, IBM/Microsoft Operating System 2, Mac OS
- Hardware Platform: Mainframe computers, Midrange computers, RISC processors, Mobile device architecture, Mac systems.
- Browsers: Internet Explorer, Firefox, Google Chrome, etc.

See also Part 2, Chapter 1, example 8.

Notes:

Currently the platforms considered in the calibration of the model are only software type platforms.

Please note that this category should be used only if same set of functionality is being delivered on multiple platforms. This is the case where business functionality is the same but it needs to be delivered in two different environments. For example, same application functions are built on JAVA and also on VC++ to suit client requirements, and then this category can be used.

If the architectural framework itself consists of different platforms to deliver part of functionality, then this category should not be used. This is a usual case where different technical components interact with each other to deliver application functions. No duplication of effort takes place to rebuild the same functionality in different environment.

3.2 Database Technology

Definition Features and operations that are added to the database or to the statements to read / write data to and from the database to deliver non-functional requirements without affecting the functionality that is provided

SCU The elementary process

Terms

Database Changes

Each of the following sub items is considered as one change.

1. Creating or changing a Business table or a Reference table, such as:
    a. Adding tables or adding columns for non-functional purposes only.
    b. Rearranging the order of column in a table.
    c. Changing or adding relationships using referential integrity features.
    d. Changing the primary key without dropping and adding the primary key.
2. Creating or updating Code Data table
    a. Adding tables or adding columns for non-functional purposes only.
    b. Rearranging the order of column in a table.
    c. Changing the primary key without dropping and adding the primary key.
3. Adding, deleting or changing an index, such as:
    a. Changing the columns used for indexing.
    b. Changing the uniqueness specification of an index.
    c. Clustering the table data by a different index.
    d. Changing the order of an index (ascending or descending).
4. Adding or changing database views (see definition in Part 1 chapter 5) and partitions, such as:
    a. Changing or adding database partitioning.
    b. Adding, changing or removing a database view.
5. Changing database capacity, such as:
    a. Tables space.
    b. Enhancing the performance features.
6. Changing a query or insert, such as:

 BY-NC-SA

a. Changes to queries or data selection or inserts to the database without adding, deleting or changing functionality.

For example, Changing the primary key and adding relationship is counted as one change.

Complexity Parameters:

1. Logical File complexity.
2. The number of database-related changes.

Changes to the database might be done for any non-functional requirement such as performance, capacity management, data integrity etc. Complexity of implementing any such change would depend on the complexity of the Logical File as well as the # of changes.

a. Logical File Complexity Factor.

		DETs		
		1-19	**20-50**	**>50**
RETs	**1**	Low	Low	Average
	2-5	Low	Average	High
	>5	Average	High	High

Table 1-19 Logical File Complexity, Database Technology

SNAP Points calculation

	FTR Complexity Factor		
	Low	**Average**	**High**
SP=	6* # of changes	9* # of changes	12* # of changes

Table 1-20 SNAP size, Database Technology

If there are multiple FTRs being impacted for the NFR which are all impacting the same EP, then the higher complexity of the FTR should only be considered as the Complexity Factor not the individual FTRs separately.

Note: Use this sub-category for new development / new requirement as well as enhancement. For a new development or new requirement, separate the requirement into its functional aspects and its non-functional aspects (See table 3-2 for more details.)

 BY-NC-SA

Example

An EP "Create Order" is designed for performance improvement. To achieve this, a "read only" database view is created on "Customer" FTR having 18 DET and 3 RET (FTR complexity is "Low")

In addition, an index is created on "Order Placed" FTR having 30 DET and 3 RET (FTR complexity is "Average").

The highest FTR complexity is "Average"; therefore, for the two changes,

SP=9*2 = 18

See also Part 2 Chapter 1 examples 3 and 9.

3.3 Batch Processes

Definition

Batch jobs that are not considered as functional requirements (they do not qualify as a transactional function) can be considered in SNAP. This sub-category allows for the sizing of batch processes which are triggered within the boundary of the application, not resulting in any data crossing the boundary.

Non-functional requirements (NFR) associated with batch jobs such as improving the job completion time, increasing the capacity of the job to process higher volumes of transactions, or performance improvement requirements may be sized using other SNAP sub-categories as applicable (3.2, 1.1 or 1.2).

However, if a NFR related to batch processing is not covered under these sub-categories, it may be considered in 3.3.

SCU

User identified batch job

Note:

When several batch jobs are automated (run always as a whole) and only the end result is user identifiable, count these batch jobs as an individual SCU

Complexity Parameters:

1. Number of DETs processed by the job.
2. Number of FTRs either read or updated by the job.

SNAP Points calculation

For each job calculate size as:

Identify the complexity level based on the number of FTRs read/updated. Calculate size as per the table below:

	Complexity Level		
	Low (1-3 FTR)	**Average (4-9 FTR)**	**High (10+ FTR)**
SP =	4*#DETs	6*#DETs	10*#DETs

Table 1-21 SNAP sizing for Batch Processes

Notes:

User specified one-time data loads to logical tables, can be counted using this category. Please note that these data loads should not be migration related data loads, which are counted as Conversion FP using function points.

Example

- Different processes are merged to one batch: Count the DETs and FTRs in the merged batch.
- Intermediate data for job validation that are in Code Data.
- Scheduler data instructs to perform subsequent process steps, which are in Code Data.

See also Part 2, Chapter 1, example 10.

 BY-NC-SA

Category 4: Architecture

Architecture The Architecture Category relates to the design and coding techniques utilized to build and enhance the application. It assesses the complexities of modular and/or component based development.

4.1 Component Based Software

Definition Pieces of software used within the boundary of the assessed application to integrate with previously-existing software or to build components in the system.

SCU The elementary process.

Terms

A Software Component

A piece of software offering a predefined service and which is able to communicate with other components via standard interfaces.
An individual software component is a software package, a Web service, or a module that encapsulates a set of related functions (or data). The essence of a "component" is the encapsulation of business logic or technical functionality which admits a standard interface. Software component is the element that conforms to a component model and can be independently deployed and composed without modification, according to a composition standard. A component model defines specific interaction and composition standards. A component model implementation is the dedicated set of executable software elements required to support the execution of components that conform to the model.

Criteria for software components:

1. Performs a specific functionality.
2. Capable of Parallel execution: multiple-use.
3. Exchangeable: Non-context-specific.
4. Composable with other components (can be selected and assembled in various combinations to satisfy specific user requirements).
5. Encapsulated i.e., non-investigable through its interfaces.
6. A unit of independent deployment and versioning with well-defined interfaces and communicates via interfaces only.
7. Has a structure and behavior that conforms to a component model like .COM, CORBA, SUN Java etc.

 BY-NC-SA

Examples Below picture shows simple components interacting with each other (source: Wikipedia[3]).

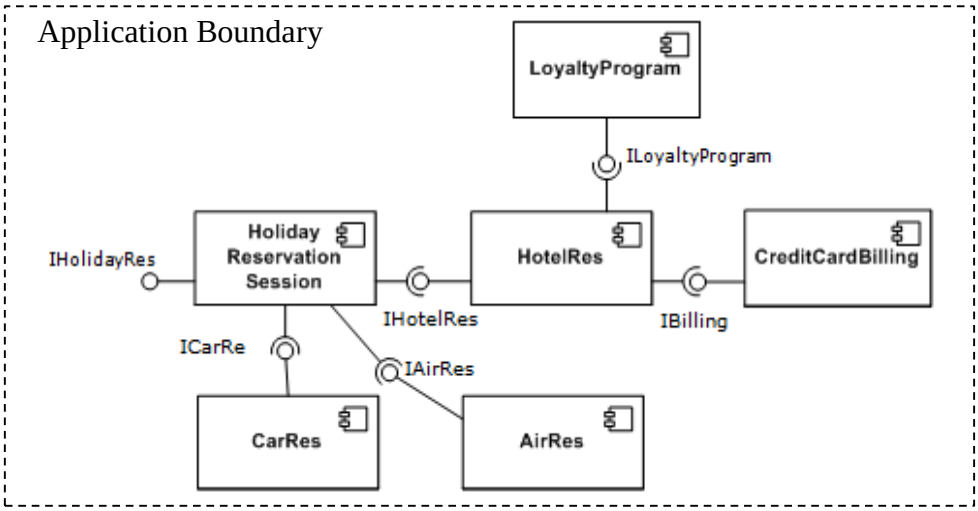


Figure 1-5 Components model for holiday reservation system

Complexity Parameters:

1. Third-party component or in-house reuse.
2. Number of unique components that are involved in the elementary process.

SNAP Points calculation Calculate size based on the constant factor and the number of unique components.

Type	SNAP Points Calculation
In-house components	SP=3*(# of unique components)
Third party components	SP=4*(# of unique components)

Table 1-22 SNAP sizing for Component based software

Example See Part 2, Chapter 1, Example 11.

Notes: This sub-category does not size the functionality of the component. Follow the instructions of the CPM to count the functional size of the components.

Reuse of components may be applied to meet non-functional requirements **such as Maintainability** ("The capability of the software product to adhere to standards or conventions relating to maintainability"), changeability, maturity or replaceability.

[3] http://en.wikipedia.org/wiki/Component-based_software_engineering

4.2 Multiple Input / Output Interfaces

Definition

Applications required supporting multiple input and output interfaces (user files with the same format) are covered in this subcategory. For example: due to a growing number of users and volume of data over a period of time.

Adding more input/output interfaces without changing the functionality is not considered functional change and hence such changes are not sized by FP. This sub-category should be used to size such changes in an application.

Note:

If the project/organization considers adding new input/output interfaces as a functional change, then function points would be used for sizing, and SNAP should not be used.

SCU

The elementary process.

Complexity Parameters:

1. The number of data element types (DET's) in the SCU.
2. The number of additional input and output interfaces.

Notes:

Count the number of additional input and output interfaces.

.

SNAP Points calculation

Identify the complexity based on number of DETs in the SCU.

The size is the product of the factor derived from the number of DETs specified in the table below and the number of added interfaces.

	Complexity Level		
	Low	**Average**	**High**
	1-5 DETs	**6-19 DETs**	**20+ DETs**
SP =	3* (Additional # of Interfaces)	4* (Additional # of Interfaces)	6* (Additional # of Interfaces)

Table 1-23 SNAP sizing for Multiple Input /Output Interfaces

BY-NC-SA

Examples

Example 1: Adding interfaces to external application A1 without adding or changing the functionality.

One SCU shown, data flows to and from the boundary.

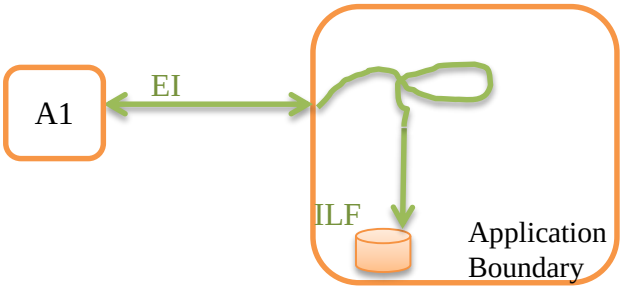


Figure 1-6.1 – Example 1, before the change

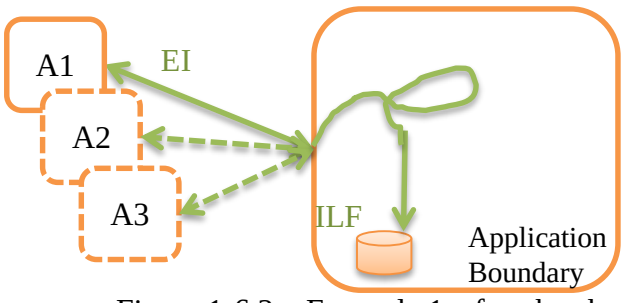


Figure 1-6.2 – Example 1, after the change

Note: The dotted lines in figures 1-6.2 to 1-6.7 indicate the change to the existing configuration.

- The number of DETs in the EP = 6 (Average complexity)
- Two additional interfaces
- SP = 4 * 2 interfaces = 8

Example 2: After the change in Example 1 was delivered, it is required to add another interfaces to external applications A1, A2, A3.

One SCU shown, data flows to and from the boundary.

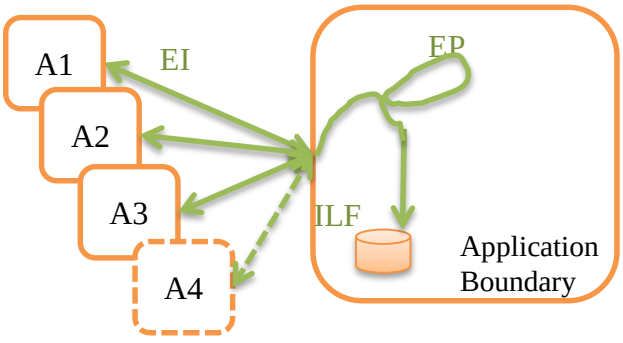


Figure 1-6.3 – Example 2, after the second change

- The number of DETs in the EP = 6 (Average complexity)
- One additional interfaces
- SP = 4 * 1 interfaces = 4

Example 3: Adding interfaces to external applications A1 and B1 without adding or changing the functionality

Two SCUs shown, data flows to and from the boundary

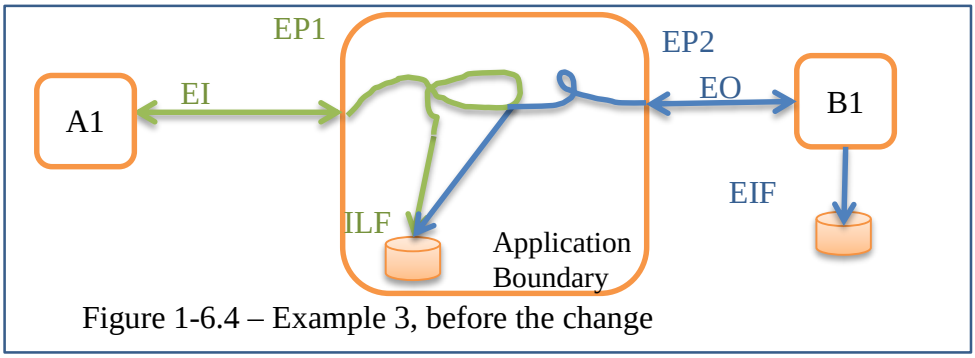


Figure 1-6.4 – Example 3, before the change

- The number of DETs in EP1 flowing to and from application A1= 5 and the number of DETs in the EP2 flowing to and from application B1= 8.

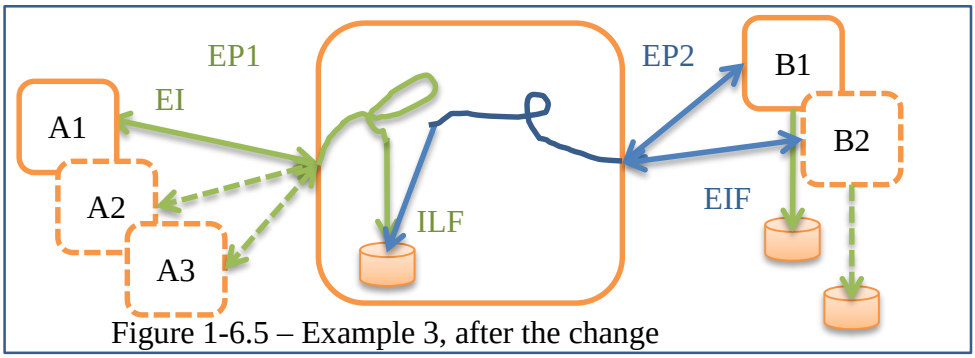


Figure 1-6.5 – Example 3, after the change

- SCU 1 = EP1
  - 5 DETs = Low
  - 2 Additional interfaces
  - SP= 3*2interface = 6
- SCU 1 = EP1
  - 8 DETs = Average
  - 1 Additional interface
  - SP= 4*1 interface = 4
- SP = 6 + 4 = 10

Example 4: Adding interfaces to external application A1 without adding or changing the functionality, two SCUs

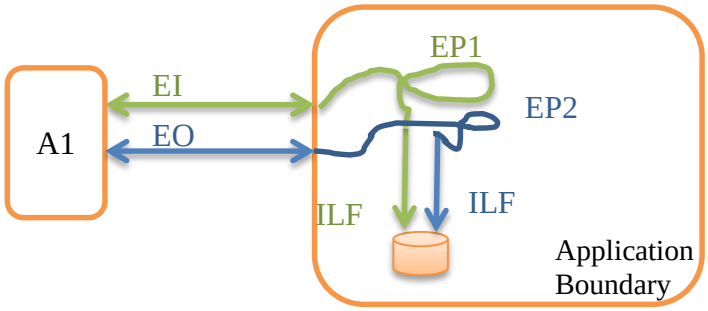


Figure 1-6.6 – Example 4, before the change

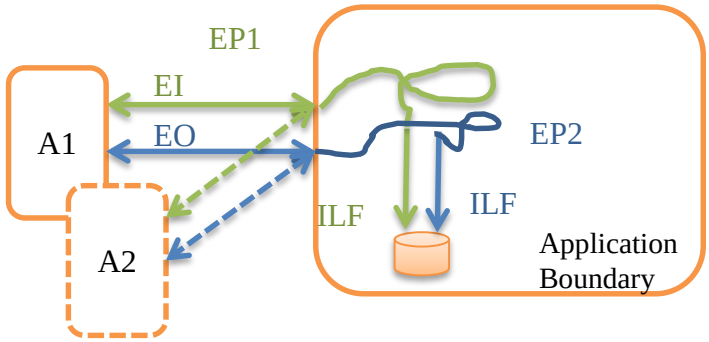


Figure 1-6.7 – Example 4, after the change

- The number of DETs in the EI = 5 and in the EO = 10
- SCU 1 = EP1
  - 5 DETs = Low
  - 1 Additional interface
  - SP= 3*1interface = 3
- SCU 2 = EP2
  - 10 DETs = Average
  - 1 Additional interface
  - SP= 4*1 = 4
- Total SP = 3 + 4 = 7

Note

A key difference between this sub-category and 2.3 / 2.4 multiple input /output methods is that in 4.2, the existing interface is replicated with the same technology to give all users the same level of performance and same experience.

This sub-category is not related to graphical user interfaces

See also Part 2, Chapter 1, example 12.

Mission Critical/Real Time Systems

Introduction

Some software systems might be tagged as Real Time or Mission Critical based on the timeliness or accuracy or severity of the consequences associated with their output.

Terms

Real Time

Real Time software is software which fails if a timing deadline is not met.

Mission Critical

Mission Critical software is a software whose failure might cause catastrophic consequences (such as someone dying, damage to property, severe financial losses, etc.)

The timeliness, accuracy, high throughput aspects of such transactions might be considered part of the functional aspect as these characteristics are the sole basis of classifying a system as Real Time/Mission Critical.

However in case the timeliness, accuracy, high throughput aspects are considered as non-functional requirements, their sizing may be done using some of the other sub-categories.

Examples:

Timeliness might be achieved by tuning the database interaction transactions or making database changes or combination of both for improved performance. In such a case, subcategory 3.2 will be used for SNAP calculation.

SNAP Points calculation

Accuracy might be achieved by adding more validations and logical/mathematical operations. Subcategory 1.1, 1.2 might be used for SNAP calculation.

Higher Throughput might be achieved by splitting existing transactions to process multiple inputs in parallel. Subcategory 2.3 might be used for SNAP calculation.

The solution mentioned for the above three scenarios is not exhaustive. If some other approach is used to meet such requirements the appropriate sub-category may be used.

Hence Real Time or Mission Critical transaction/system should be assessed by using the other sub categories.

 BY-NC-SA

SNAP Calculation Example

The following table demonstrates how SNAP size is calculated for the "Data Operations" category.

The complexity is calculated according to the formulas presented in this Chapter.

Note: This example is shown using the template of SNAP counting tool. This tool is available on IFPUG's online store.

		Category 1: Data Operations															
	Sub-Category:	1.1 Data Entry Validation					1.2 Logical and Mathematical Operations						1.3 Data Formatting				
	SCU Definition:	SCU=The Functional Elementary Process					SCU=The Functional Elementary Process						SCU=The Functional Elementary Process				
No.	SCU Description / ID	Number of nesting levels	Complexity Level	Number of DETs	Formula	Data Entry Validation SNAP Count	Elementary Process Type (Main Purpose)	Number of DETs	Number of FTR	Complexity Level	Formula	Logical and Mathematical Operations SNAP Count	Transformation Complexity Level (L, A, H)	Number of DETs	Formula	Data Formatting SNAP Count	
1	Customer orders a product	2	Low	18	=2*# DETs	36	Logical	4	1	Low	=4*# DETs	16	Low	4	=2*# DETs	8	
2	Customer amends an order																
3	Invoice details changes	4	Average	15	=3*# DETs	45	Mathematical	12	2	Low	=3*# DETs	36					
4	Validate address	6	High	9	=4*# DETs	36											
Totals						117						52				8	

	Sub-Category:	1.4 Internal Data Movements				Internal Data Movements SNAP Count	1.5 Delivering Added Value to Users by Data Configuration				Delivering Functionality by Data Configuration SNAP Count	Total SP for Data Operations:
	SCU Definition:	SCU=The Functional Elementary Process					SCU=The Functional Elementary Process					
No.	SCU Description / ID	Number of FTRs	Complexity Level	Number of DETs	Formula		Number of Records	Complexity Level	Number of Attributes	Formula		
1	Customer orders a product	3	Low	18	=4*# DETs	72	9	Low	7	=6*# Attributes	42	174
2	Customer amends an order						9	Low	7	=6*# Attributes	42	42
3	Invoice details changes											81
4	Validate address	1	Low	4	=4*# DETs	16						52
						88					84	

Table 1-24 SNAP Calculation Example

 BY-NC-SA

This page intentionally left blank

 (cc) BY-NC-SA

Calculate Non-functional Size (SNAP Points)

Introduction This section defines how to calculate the non-functional size of the project/product in scope.

Contents This chapter includes the following:

Topic	Page
Formula Approach	6-2
Determine the Non-functional Size of Each Sub-category	6-2
Determine the Non-functional Size for a Development Project	6-2
Determine the Non-functional Size for an Enhancement Project	6-3
SNAP Calculation Case Study	6-6

 (cc) BY-NC-SA

Formula Approach

During an assessment of a non-functional requirement, one or many of the sub-categories can be assessed depending on the specification of the requirement. For each non-functional requirement, it is possible to determine the non-functional size in four steps:

Step 1 For each requirement, identify the categories and sub-categories that are associated with the requirement.

Step 2 For each of the sub-categories, identify the SCUs

Step 3 Determine the non-functional size (SNAP Points -SP) for each SCU within the sub-category, by using the equation or the table for the sub-categories.

Step 4 Determine the SNAP Points for a specific project or application by using the formula for the project type in question.

Determine the Non-functional Size of Each Sub-category

The non-functional size of each sub-category shall be determined using the defined measure for the SCU for each sub-category.

There is one definition of the SCU for each of the sub-categories. These assessment criteria are defined in the sub-category definition in Chapter 2.

The size for each sub-category is determined by using the defined equation or table for each sub-category.

Determine the Non-functional Size for a Development Project

The size of the non-functional requirements is equal to the sum of SP sizes of each category.

A development project non-functional size shall be calculated using the development formula

Formula for development project

DSP = ADD

where

DSP is the development project SNAP size;

ADD is the size of the non-functional requirements delivered to the user by the development project.

ADD = $\sum$ of SP for all sub-categories

The application non-functional size is equivalent to the non-functional size of the development project.

Note: For non-functional size converted functionality is not identified.

 (cc) BY-NC-SA

Determine the Non-functional Size for an Enhancement Project

Enhancement projects can involve additions, changes to, and deletion of existing non-functional features.

Enhancement project is a project to develop and deliver maintenance. It may be adaptive, preventive or perfective type

The enhancement project non-functional size is a measure of the non-functional characteristics added or deleted at the completion of an enhancement project as well as changes made to existing non-functional characteristics as measured by the enhancement project SNAP size.

Rules

Enhancement non-functional requirements shall be measured in accordance with the following:

a) Do not modify the boundary or partition already established for the application(s) being modified

b) Assess requirements that are added, changed or deleted

c) The application non-functional size may be updated to reflect:

1) Added non-functional requirements, which increase the application non-functional size.

2) Changed non-functional requirements, which may increase, decrease or have no effect on the application non-functional size.

3) Deleted non-functional requirements, which decrease the application non-functional size.

Note:

This rule is analyzed and counted per each sub-category

Formula for enhancement project

An enhancement project non-functional size shall be calculated using the formula:

$$ESP = ADD + CHG + DEL$$

where

ESP is the enhancement project SNAP size.

For each sub-category,

ADD is the size of the non-functional characteristics being added by the enhancement project.

CHG is the size of the changes made to existing sub-categories by the enhancement project.

DEL is the size of the non-functional characteristics deleted by the enhancement project.

For a sub-category SC,

$$ESP_{SC}=ADD_{SC}+CHG_{SC}+DEL_{SC}$$

 BY-NC-SA

For the enhancement project:

ASP = ∑of ESP for sub-category 1.1 + ∑of ESP for sub-category 1.2+……∑of ESP for sub-category 4.2

(∑ = Sum)

An application non-functional size after an enhancement project shall be calculated using the formula:

Formula for application after enhancement project

ASPA = ASPB + (ADD + CHGA) - (CHGB + DEL)

Where:

ASPA is the application SP after the enhancement project.

ASPB is the application SP before the enhancement project.

ADD is the size of the non-functional requirements being added by the enhancement project.

CHGA is the size of the non-functional requirements being changed by the enhancement project – as they are/will be after implementation.

CHGB is the size of the non-functional requirements being changed by the enhancement project – as they are/were before the project commenced.

DEL is the size of the non-functional requirements deleted by the enhancement project.

Note:

The size of changes made to sub-categories in an enhacement project (i.e., CHG) does not provide an assessment of the overall size of the sub-categories before (CHGB) or after (CHGA) the enhancement project – CHG measures the size of the change in the non-functional charcteristics being changed.

Example 2 below demonstrates a situation in which the size of the change in the project (CHG) is 12 SP, but the overall size of the sub-category is 0 (CHGA = CHGB)

Therefore, in order to maintain the application's non-functional size after each enhancement project, one must assess, in addition to the size of the changes made, the size of the sub-category before and after the enhancement.

 (cc) BY-NC-SA

Examples

1. An application has an elementary process in which there are 10 DETs. Three DETs are already encrypted (encryption complexity is average).

   The enhancement project requires to change the encryption type (new encryption is local).

Counting the enhancement project:

Type of requirements: CHG

Transformation complexity: High

ESP = ADD + CHG + DEL

= 0 + 5*# of DETs + 0 = 15

Counting the application:

ASPB =3*# of DETs= 9 (Assuming this is the only NFT in the application)

CHGB = 3*# of DETs= 9

CHGA = 15

ASPA = ASPB + (ADD + CHGA) - (CHGB + DEL)

= 9 + (0 + 15) - (9 + 0) = 15

2. A "Search" application has an entry screen with one entry: a string to search. To keep the look-and-feel of the application fresh, each month the background color is changed, the size and location of the search field are changed and the shape and location of the search button are changed.

   To build this screen, the sum of the number of unique properties is 20.

Counting the enhancement project:

Type of requirements: CHG

UI Type complexity: High

Number of unique UI elements: 3 (Screen, field, control);

ESP = ADD +CHG+ DEL

= 0 + 4*3 + 0 = 12

Counting the application:

ASPB =3*4= 12 (Assuming this is the only NFT in the application)

CHGB = 3*4= 12

CHGA = 3*4= 12

ASPA = ASPB + (ADD + CHGA) - (CHGB + DEL)

= 12 + (0 + 12) - (12 + 0) = 12

(No change in application size)

 (cc) BY-NC-SA

SNAP Calculation Case Study

Project scope

'StarTrek' is the code name for an enhancement project that the 'Alpha' team was commissioned to deliver. Being a relatively small project (in terms of budget), StarTrek had requirements to enhance the performance of the company's flagship application, 'Centra-One', along with minor changes to some of its existing functionalities.

SNAP Meeting

The brainstorming sessions held between the project's Chief architect, the Lead developer and the client resulted in the following set of requirements:

In order to boost the performance of the "invoice update" transaction (current average response time: 8-12 seconds), and bring the response time down under 3 seconds, a multi-pronged strategy was adopted.

Requirements Relevant for SNAP

Requirements relevant for SNAP are identified and reviewed

Requirement 1:

Create an additional interface for receiving "update invoice" transactions – Automated Fax interface (to divert a portion of the web update transactions and lessen the load that the server was facing). A separate server was installed to handle fax requests and was equipped to read and decode the incoming fax forms which were in a predetermined format.

Requirement 2:

a) Modify the validation logic of the current 'update invoice' online transaction to reduce its processing time.
The new validation needs one additional DET. This is considered as a functional change, generating a High EI (3 FTRs, 10 DETs)
(The validation is considered as a non-functional change)

b) Create a cache file that stores the list of most-commonly accessed customer records, which was internally referenced before updating the invoice. This, it was predicted, will result in further improvement in performance. This cache was refreshed at regular intervals.

c) Make some cosmetic changes to the 'update invoice' online screen and the 'home screen' to inform the users of the new setup.

The Alpha team wanted to size these requirements, to provide as an input to the estimation process and also to track the size change over the life cycle to indicate scope-creep. They naturally decide to use SNAP together with IFPUG FP as that allowed them the capability to size all functional and non-functional aspects of the project.

 BY-NC-SA

Purpose & Scope

Purpose of the sizing: to size the enhancement project and use it for project estimation and to track scope creep.

Type of count: Enhancement

Scope: Requirements 1 & 2 (a, b & c)

Boundary

SNAP assessment will assume the same boundary as that used by the IFPUG FP methodology. Though the Centra-One application had 3 layers in itself, the boundary was fixed at the same level as that for the FP methodology.

Assessment

During the high level design phase, the Alpha team did an impact analysis and found that the following SCUs would be impacted. The SCUs were determined as per the rules of SNAP.

1. Identify sub-categories
2. Identify the SCUs
3. Per sub-category, assess the complexity of each SCU, and calculate SNAP score.

Notes:

One requirement that includes both functional and non-functional aspects may impact both function points and SNAP (See 2a above)

One requirement may impact more than one elementary process and one SCU can be impacted by more than one requirement (functional/non-functional). While assessing, care must be taken to ensure that the SCUs are not double assessed in SNAP.

The definition of SCUs according to SNAP may be identical, for some sub-categories, to the traditional IFPUG elementary process. In other cases, sub-categories are assessed within different counting units (SCUs) level.

In addition to IFPUG FP counting, which has determined a count of 6 FP to this project, SNAP assesses it as 200 SP (38+34+31+41+24+32) based on the analysis in table 1-25 below.

 BY-NC-SA

The Alpha team used both the functional size (6 FP) and the non-functional size (200 SP) as input into their estimation process.

Req. #	*SCU Type*	Impacted SCU	FP counted	# of FP	SNAP counted	1.1 Data Entry Validation	1.3 Data Formatting	2.1 User Interfaces	2.3 Multiple Input methods	3.2 Database Technology
1	Elementary process	Update invoice	No	0	Yes	-	30	-	4	-
2a	Elementary process	Update invoice	Yes	6	Yes	8	-	-	-	-
2b	Elementary process	1. View customer 2. Update customer details	No	0	Yes	-	-	-	-	18
2c	Elementary process	1. Updates Invoice 2. Home screen	No	0	Yes	-	-	25	-	-

Table 1-25 SNAP Case Study

Part 2 – Examples

This page intentionally left blank

 (cc) BY-NC-SA

Examples

Introduction This chapter includes several examples of how non-functional requirements can be assessed using the categories and sub-categories.

Contents This chapter includes the following examples:

Topic	Page
Example 1: Change Free Text to Valid Values	1-2
Example 2: Data Security	1-3
Example 3: Internal Data Backup and Data Transformation	1-4
Example 4: Creating New Products and Offers	1-6
Example 5: Compliance with Standard	1-8
Example 6: Help	1-10
Example 7: Adding Input and Output Methods	1-11
Example 8: Multiple Software Platforms	1-13
Example 9: Performance Improvement	1-14
Example 10: Batch Jobs	1-16
Example 11: Using Components	1-17
Example 12: Multiple Interfaces	1-18

 (cc) BY-NC-SA

Example 1: Change Free Text to Valid Values

Requirement: An international retail store ordering system has several free-text fields to be replaced by lists of valid values to improve accuracy of data entered and reduce order failures due to validation errors. No new functionality is requested.

Address validation will check country, state, county, city, street and house number. Product description is built in hierarchy of product type, manufacturer, model number and color.

Analysis: Two elementary processes were identified which were impacted by the requirement and need to be enhanced ("Customer orders a product" and "Customer amends an order").

The solution design involves one sub-category ("Data entry validation"). The SCU is the elementary process.

Address validation is done in 6 nesting levels: first country (1), then state (2), then county (3), then city (4), then street (5) and finally the house number (6). Product validation has 4 nesting levels: first type (1), then manufacturer (2), then model number (3) and then color (4).

The "Customer orders products" EP will execute both the address and product hierarchy validations. Since address validation has 6 nesting levels and product validation has 4 nesting levels, the longest chain of validations to be considered for 1.1 is 'Address validation'. Hence the number of nesting levels to be considered for customer orders product is 6, which gives high complexity.

The "Customer amends an order" EP will execute only the product hierarchy validation which has 4 nesting levels. (4 DETs are nested). Hence the nesting level for this EP will be 4, which gives average complexity.

Counting:

		1.1 Data Entry Validation			
No.	**SCU Description**	**# DETs**	**# Nesting Levels**	**Formula**	**SP=**
1	**Customer orders products**	**10**	**6**	**=4*#DETs**	**40**
2	**Customer amends an order**	**4**	**4**	**=3*#DETs**	**12**

Table 2-1: Example 1, SNAP Calculation for Data Entry Validations

Total SNAP size for the project = ∑ SP for all SCU of the sub-category = 40+12 = 52 SP

(cc) BY-NC-SA

Example 2: Data Security

Requirement: To meet the new security standards of a bank, it is decided to encrypt the data transferred from one system to another as well as the data displayed on user interfaces. Sensitive Personal Information (SPI) data should be encrypted by the application before passing on to the other systems. Any SPI data to be displayed on user interfaces should be masked by * symbol.

The design for the solution required writing a program to implement encryption of SPI data which will be used by any process in application 1.

Analysis: The solution design involves one sub-category ("1.3 Data Formatting"). The SCU is the elementary process.

Three elementary processes were identified as impacted: "View subscriber details," "View payment history" (both to mask SPI data by *) and "Customer information extract sent to application 2" (process to send 16 bit encrypted data).

To decide the transformation complexity level, masking data by * would qualify as a Low and encryption of data would qualify as high complexity. The following data is considered as SPI: credit card number, blood group, SSN, telephone number, credit history)

This information is used in all 3 processes

Counting:

		1.3 Data Formatting			
No.	**SCU Description**	**Transformation Complexity**	**# DETs**	**Formula**	**SP =**
1	View subscriber details	Low	5	2* #DETs	10
2	View payment history	Low	5	2* #DETs	10
3	Customer information extract sent to application 2	High	5	5 *#DETs	25

Table 2-2: Example 2, SNAP Calculation for Data Formatting

Total SNAP size for the project =

$\sum$ SP for all SCU of the sub-category = 10+10+25 = 45.

 (cc) BY-NC-SA

Example 3: Internal Data Backup and Data Transformation

Requirements: An application is designed as three tier solution architecture: User Interface, Middle Layer and Backend Server. Backend server layer holds the database, Middle Layer holds the business processes and User Interface is the front end for the users of the application to view and maintain data.

This application need to enhance to advanced technology platform to create a system that will be more intuitive and easier to use. The platform includes hardware and software that effectively manages connectivity, access, and synchronization. This means that User Interface should support for desktop user and remote device (Handset - as with a smart phone) for portability. To reduce the time and effort, remote device users must be able to download the latest work and access the data. This requires replicating the data from server into the handset.

1. Data on the user interface should be grouped in order to improve usability
2. Safety and recoverability: It is required to create a back-up to important data

Analysis: The three layers of the solution are considered as the three different partitions within this application boundary.

The "View Orders" process will be enhanced to do the data grouping in the middle layer partition and then pass on the data to the user interface partition. The enhancement would be done only in the middle Layer. The "View Orders" process takes a total of 20 DETs as a sum of unique input and output fields at the middle layer and it reads/updates a total of 2 ILFs/EIFs. This solution for requirement 1 involves one sub-category (1.4 Internal Data Movements). SCU is the EP within the Middle Layer partition.

The backup would be created a table in the Backend Layer and the backup process would back up the data from "Order" data file in the backend and delete the data after 1 day. This requires creating a new Back up process and creating a new Back up table within the Backend partition.

The "Order" file in the Backend is replicated in the front end for user access. All the DETs in the "Order" file should be considered here. FTRs are the backend file and the front end replicated file.

The "Back up process" will read 20 DETs from the "Order" data file and update the same in the Back up table. The solution for requirement 2 involves 2 sub categories: 1.4 (Internal Data Movements) for the new backup process, and 3.2 (Database Technology) for the new back up table.

 (cc) BY-NC-SA

Counting:

		1.4 Internal Data Movements			
No.	**SCU Description**	**Transformation Complexity**	**# DETs**	**Formula**	**SP =**
1	View Orders	Low	20	4*DETs	80
2	Back Up process	Low	20	4*DETs	80

Table 2-3: Example 3, SNAP Calculation for Internal Data Movements

		3.2 Database Technology			
No.	**SCU Description**	**FTR Complexity**	**# Changes**	**Formula**	**SP =**
1	Back up Table	Low	1	6*#changes	6

Table 2-4: Example 3, SNAP Calculation for Database Technology

Total SNAP size for the project = ∑ SP for all SCU of the sub-category = 80+80+6 = 166 SP

 (cc) BY-NC-SA

Example 4: Creating New Products and Offers

Requirements A telecom software application is designed for easy maintainability and fast launch of new products and offers for its customers. The service provider only needs to create a set of configurations (reference data) to launch a new product without any code change; No logic changes are required for processing the new orders or offers.

Ten new products are to be launched along with three new offers (An offer is a bundle of products with specific prices. An offer is limited in sale period, it can be offered for a limited period)

Attributes	Product 1	Product 2	Product 3	Product 4	Product 5	Product 6	Product 7	Product 8	Product 9	Product 10
LOB	IPTV	IPTV	Internet access	Internet access	Internet access	Land line	Land line	Land line	Mobile	Mobile
Bandwidth	5	10	5-100	20	5	(-)	(-)	(-)	(-)	(-)
# of channels	100	150	(-)	(-)	(-)	(-)	(-)	(-)	(-)	(-)
SLA	Gold/ Silver/ Platinum	Gold/ Silver/ Platinum	Gold/ Silver/ Platinum	Gold/ Silver/ Platinum	Gold/ Silver/ Platinum	Regular /Senior	Regular /Senior	Regular /Senior	(-)	(-)
# of devices	1-5	1-5	1-5	1-5	1-5	(-)	(-)	(-)	(-)	(-)

Table 2-5 Example 4, Main description of the new products

The offers' attributes

1. List of products.
2. Price plan scheme.
3. Discount.
4. Valid from.
5. Valid until.

Analysis: The design for solution required configuring 10 new products in the "Products" logical file and 3 new offers are to be created in the "Available Offers" logical file. The solution involves only one sub-category, 1.5 (Delivering Added Value to Users by Data Configuration).

Two elementary processes are impacted. "Create Order" and "Modify Order" are the two processes which would consume the product and offer configurations. (The processes of creating or modifying an order (Add/change data) do not generate SNAP Points – see notes in the definition of the SCU of sub-category 1.5). The "Products" file consists of 10 records to be configured and as there are 10 products, 10 records will be configured. For "Available Offers", 5 attributes to be configured for each offer and as there are 3 offers to be created, 3 records will be configured.

(cc) BY-NC-SA

Counting:

SCU: "Create Order"

		1.5 Delivering Added value to users by Data Configuration			
No.	**The Logical File**	**Complexity Level**	**# Attributes**	**Formula**	**SP=**
1	Products	Low (10 records)	5	6*attributes	30
2	Offers	Low (3 records)	5	6*attributes	30

Table 2-6: Example 4, SNAP Calculation for Delivering Added Value to Users by Data Configuration, first SCU

SCU: "Modify Order"

		1.5 Delivering Added value to users by Data Configuration			
No.	**The Logical File**	**Complexity Level**	**# Attributes**	**Formula**	**SP=**
1	Products	Low (10 records)	5	6*Attributes	30
2	Offers	Low (3 records)	5	6*attributes	30

Table 2-7 Example 4, SNAP Calculation for Delivering Added Value to Users by Data Configuration, second SCU

Total SNAP size for the project =

$\sum$ SP for all SCU of the sub-category = 30+30+30+30 = 120 SP

 (cc) BY-NC-SA

Example 5: Compliance with Standard

Requirements: Compliance with ADA standard 508 or W3C Web Content Accessibility Guidelines (WCAG) 2.0 for accessibility.

Add accessibility options, so that people with difficulties to hear sounds, and people with difficulties to see a normal display, can use the application easily.

The proposed design is:

- Add pop-up icons whenever a sound is generated (there are four different sounds)
- Add big and simple fonts, with one size (14 pt.) to all Menus and Fields on all screens

One specific color instead of the normal font options.

Analysis:

- Changing the font size from 10 pt. to 14 pt., changing font colors are considered as technical changes. The requirement is not considered as adding new functionality or changing the functionality, therefore no FP are generated.
- We assume that the icons do not need any computations (no animation).
- The design involves one SNAP sub-category, ("User Interfaces").
- The SCU is the elementary process.
- The new icons and font change affect 5 elementary processes (that do not overlap).
- In two elementary processes, the change is counted as 'Simple'; In 2 Processes, the change is 'Average' and one process is considered Complex.
  (Assuming < 10 GUI properties added/changed for Simple, 10 - 15 properties added/changed for Average and more than 15 properties added/changed for Complex).

There is a fixed set of UI element for each EP.

Note: The four icons that accompany the sounds are one unique UI element (8 properties: name, type, resolution, size, orientation, open width, location (x), location (y). Fonts appear in the following unique UI elements: menus, icons, up to 11 types of controls, tabs.

- EP 1 – 5 unique UI elements impacted.
- EP 2 – 10 unique UI elements impacted.
- EP 3 – 5 unique UI elements impacted.
- EP 4 – 13 unique UI elements impacted.
- EP 5 – 7 unique UI elements impacted.

Counting:

		2.1 User Interfaces			
No.	**SCU Description**	**Complexity Level**	**# of Unique UI elements**	**Formula**	**SP =**
1	EP 1	Low	5	2*# unique UI elements	10
2	EP 2	Low	10	2*# unique UI elements	20
3	EP 3	Average	5	3*# unique UI elements	15
4	EP 4	Average	13	3*# unique UI elements	39
5	EP 5	High	7	4*# unique UI elements	28

Table 2-8: Example 5, SNAP calculations for User Interfaces

Total SNAP size for the project =

$\sum$ SP for all SCU of the sub-category = 10+20+15+39+28 = 112 SP

 (cc) BY-NC-SA

Example 6: Help

Requirements: Enhancing additional "Help" to an application.

The proposed design:

a) Pop-up screens will appear when the user right-clicks on a field, with an explanation of how and when this field should be used (estimated number of fields: 60, plus 40 context-sensitive).

b) Explanation will include a hyper-link to either a video with a demonstration, or a wizard with a set of Q&A (50 links to Q&A and 15 videos).

Analysis and counting: The design involves the sub-category Help Methods. The following table shows how help items are sized:

		2.2 Help Methods			
No.	**Help Item**	**Help Type**	**# added Help Items**	**Formula**	**SP=**
1	Pop-up boxes	On-Line Text	60	=2*(#help items)	120
2	Pop-up boxes	Context + On-line	40	=3*(#help items)	120
3	Hyperlinks to Q&A	Context	50	=2*(#help items)	100
4	Hyperlinks to Videos	Context	15	=2*(#help items)	30

Table 2-9: Example 6, SNAP calculation for Help methods

Total SNAPP size for the project =

$\sum$ SP for all SCU of the sub-category = 120+120+100+30 = 370 SP.

Example 7: Adding Input and Output Methods

(Single Instance approach)

Requirements

A banking software application supports five different processes (in FP terms elementary processes): Create Account, Modify Account, Make Payment, End-of-Day (EOD) Account Creation summary report, EOD Credit Debit report

At present, the three elementary processes of 'Create Account', 'Modify Account' and' Make Payment', take input by keying in data from the keyboard. The bank wants to enhance the software to be able to accept input for these three processes in the form of scanned documents and by reading a barcode as well.

(The "Create Account" and "Modify Account" processes 20 DETs each, and "Make Payment" process processes 15 DETs).

The "EOD Account Creation Summary Report" and "EOD Credit Debit Report" are currently sent out in printed CSV format. The bank wants to enhance the software to be able to produce the output for these processes in the form of printed PDF as well as inline mail to the recipients.

(The EOD Account Creation Summary Report has 15 DETs and EOD Credit Debit Report has 10 DETs).

Analysis and counting:

The design solution for this requirement involves two subcategories 2.3 (Multiple Input Methods) and 2.4 (Multiple Output Methods).

		2.3 Multiple Input Methods			
No.	**SCU Description**	**Complexity Level**	**# Additional Input Methods**	**Formula**	**SP =**
1	Create Account	High	2	6* # Additional Input Methods	12
2	Modify Account	High	2	6* # Additional Input Methods	12
3	Make Payment	Average	2	4* # Additional Input Methods	8

Table 2-10: Example 7, SNAP calculation for Multiple Input methods

 (cc) BY-NC-SA

		2.4 Multiple Output Methods			
No.	**SCU Description**	**Complexity Level**	**# Additional Output Methods**	**Formula**	**SP =**
4	EOD Account Creation Summary Report	Average	2	4* # Additional Output Methods	8
5	EOD Credit Debit Report	Average	2	4* # Additional Output Methods	8

Table 2-11: Example 7, SNAP calculations for Multiple Output methods

Total SNAP size for the project =

$\sum$ SP for sub-category 2.3 + $\sum$ SP for sub-category 2.4= 12+12+8+8+8=48

Example 8: Multiple Software Platforms

Requirements

While deciding the solution for a software project it was decided that part of it would need to be developed in a multi-platform environment.

The system consists of three tiers. The front-end tier is developed using Java; the middle tier is developed using C++ and the third tier uses SQL.

To ensure that the application can become compatible with different devices, some of the front-end functionality needs to be developed also on a different platform, using XML.

Three elementary processes (EP1, EP2, and EP3) in the application are using all three tiers. (Each is developed as a combination of Java, C++ and SQL)

Two additional processes (EP4, EP5) are doing front-end and middle processing and they do not use the new XML part. (they are developed with a combination of Java and C++)

Analysis:

The solution involves sub-category 3.1 Multiple platforms, but not for all involved EPs:
When the architectural framework itself consists of different platforms to deliver part of the functionality, then sub-category 3.1 should not be used. Therefore, EP4 and EP5 are not qualified to have SNAP size using this sub-category.

Moreover, the need for developing the software in multiple software platforms is a technical requirement; hence, it is not covered in FP.

This solution involves subcategory 3.1 Multiple Platforms.

Counting:

		3.1 Multiple Platforms		
No.	**SCU Description**	**# of Software platforms**	**Same software family?**	**SP =**
1	EP1	2 (XML, JAVA)	No	40
2	EP2	2 (XML, JAVA)	No	40
3	EP3	2 (XML, JAVA)	No	40
4	EP4	1	N/A	None
5	EP5	1	N/A	None

Table 2-12: Example 8, SNAP calculation for Multiple Platforms

Total SNAP size for the project = ∑ SP for all SCU of the sub-category = 40+40+40= 120 SP.

 (cc) BY-NC-SA

Example 9: Performance Improvement

Requirements

The customer of Telecommunication software applications requires improving the performance of some functionality.

The throughput time to create order and create subscriber need to be improved from average of 2 minutes to 1.5 minutes or less. "View payments" should be improved from 10 seconds to 8 seconds or less for all customers. "Make Payment" transactions need to be improved from 3 seconds to 2 seconds

Analysis

The design of the solution required the following changes:

1. Tuning SQL queries in "Create Order", "Create Subscriber" and "Make Payment", to make database updates faster. (Using parameterized SQL, improving database connection handling, wise implementation of DB commits etc.)
2. Creation of an indexed view on "Payments" database file, so that "View Payments" can read the view instead of reading the database. (It will improve the performance of "View Payments" process). Corresponding changes in the SQL queries need to be made to read from the new view.
3. The "Create Order" and "Create Subscriber" processes read/update Customer, Subscriber and Order database files. Customer and Subscriber database files have more than 5 RETs and more than 50 DETs, therefore are High complex. The "Order database" file is Average complexity (~30 DETs).
4. "Make Payment" reads/updates Subscriber and Payments database files. Payments database file is an Average complexity FTR; Subscriber FTR is High, therefore the FTR complexity is "High."
5. View Payments reads Payments database file only.

The solution for this requirement involves one sub-category: 3.2 Database Technology.

 BY-NC-SA

Counting:

		3.2 Database Technology					
No.	**SCU Description**	**Highest complexity level of FTRs involved**			**# of database changes**	**Formula**	**SP =**
		# of DETs	**# of RETs**	**Complexity**			
1	Create Order	>50	5	High	1	12*# of Changes	12
2	Create Subscriber	>50	8	High	1	12*# of Changes	12
3	Make Payment	50	5	High	1	12*# of Changes	12
4	View Payment	25	5	Average	2	9*# of Changes	18

Table 2-13: Example 9, SNAP calculation for Database Technology

Total SNAP size for the project = ∑ SP for all SCU of the sub-category = 12+12+12+18 = 54 SP

 (cc) BY-NC-SA

Example 10: Batch Jobs

Requirements

A banking software application provides functionalities to accept all deposits and payments and functionalities to withdraw cash, transfer money and make payments.

The Bank has a requirement to create an EOD (End-of-Day) batch job, which will:

1. Read the following logical database files: Account, Payments and Credits;
2. Apply the said business logic for required calculations and data transformations; and
3. Update two logical database files which are: Credit Summary and Debit Summary.

Analysis:

The batch job is triggered at midnight and processes all the data of the past 24 hours. This batch job doesn't take any input business data and doesn't give out any output business data. The entire scope of the job is limited within the application boundary.

The batch job should read 3 DB files and update 2 DB files. The job processes a total of 25 DETs.

Since there is no input or output crossing the boundary, implementing FP might not be possible. Hence, this requirement involves 1 sub-category: 3.3 Batch Processes.

Counting:

		3.3 Batch Jobs				
No.	**SCU Description**	**# of FTRs**	**Complexity Level**	**# of DETs**	**Formula**	**SP=**
1	Credit Debit Daily Summary Batch Job	5	Average	25	6*# of DETS	150

Table 2-14: Example 10, SNAP calculation for Batch Processes

Total SNAP size for the project = 6*25 = 150 SP

BY-NC-SA

Example 11: Using Components

Requirements: A retail customer approaches a software vendor to create a new retail shopping website designed for new target buyers – teen agers (In addition to the web site designed for the traditional consumers). After analyzing the requirements of the customer the vendor lists down different components already developed by the customer for other web sites, and can be used in the current application. The components required are:

1) Login.
2) Display inventory.
3) Compare products.
4) Add and store in shopping cart.
5) Capture customer details.
6) Capture shipping details.
7) Make payment.

Analysis: The analysis by the vendor team concludes that they can reuse components 1, 4 and 7 from their in-house product offerings without any customization. The remaining functionalities will need to be newly developed / tailored as per the needs of the customer.

The project re-uses 3 in house components.

Assuming one elementary process is involved, the SNAP assessment for component based software development is:

Counting:

Type	SP Calculation
In-house components	SP=3*3 = 9
Third party components	N/A

Table 2-15: Example 11, SNAP calculation for Component Based Software

Total SNAP size = ∑ SP for the in-house component = 9 SP

 (cc) BY-NC-SA

Example 12: Multiple Interfaces

Requirements:

A Telecom Rating software application currently receives input from one Network application and sends output to a Billing application.

After acquiring another company (with different applications), the Telecom company has decided to merge the Rating activities into its Rating application.

Calls and Data Usage information should flow to the Rating application from additional two inputs. The Rating application should send all voice and data usage information to additional output.

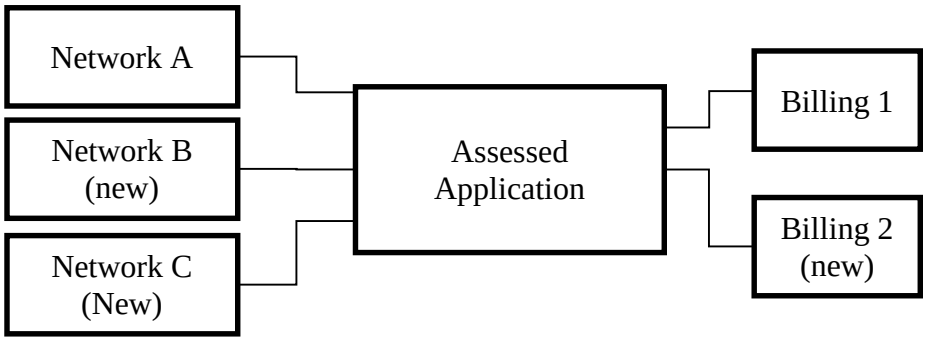


Figure 2-1: Example 12, Required Architecture View

Analysis:

At present, the "Voice usage extract" receives input from External Application and sends the output to Billing. After the change is implemented, it needs to verify the input interface and - based on the input interface - send the output to the corresponding Billing system, after implementing any interface-specific formatting or sorting of the data. The functionality would remain the same and there is no functional enhancement required.

Assuming the Voice Usage Extract and Data Usage Extract takes 20 DETs as input and output.

The design solution involves one sub-category: 4.2 Multiple Input / Output Interfaces

We need to calculate the SP for Input and Output interfaces separately.

(cc) BY-NC-SA

Counting:

Input Interfaces:

No.	SCU Description	4.2 Multiple Input /Output Interfaces			
		Complexity level based on # DETs	**# of additional input Interfaces**	**Formula**	**SP =**
1	Voice Usage Extract	High	2	6 * (Additional # of Interfaces)	12
2	Data Usage Extract	High	2	6 * (Additional # of Interfaces)	12

Table 2-16: Example 12, SNAP calculation for Multiple Input Interfaces

Output Interfaces:

No.	SCU Description	4.2 Multiple Input/ Output Interfaces			
		Complexity level based on # DETs	**# of additional output Interfaces**	**Formula**	**SP =**
1	Voice Usage Extract	High	1	6 * (Additional # of Interfaces)	6
2	Data Usage Extract	High	1	6 * (Additional # of Interfaces)	6

Table 2-17: Example 12, SNAP calculation for Multiple Output Interfaces

Total SNAP size for the project = ∑ SP for the inputs and outputs = 12+12+6+6 = 36 SP

 (cc) BY-NC-SA

This page intentionally left blank

Part 3 – Appendices

 (cc) BY-NC-SA

This page intentionally left blank

(cc) BY-NC-SA

Part 3 - Appendices

Introduction

Part 3 provides Appendices on several related topics.

Appendix A provides a Glossary of terms used within the SNAP process and the APM.

Appendix B provides examples of usage and linking between functional and non-functional sizes.

Appendix C provides a document Index.

Appendix D details the APM changes from each previous version

Contents

Part 3 includes the following sections:

Topic	Page
Appendix A – Glossary	A-1
Appendix B – IFPUG APM Link to IFPUG CPM	B-1
Appendix C – Index	C-1
Appendix D - APM Changes	D -1

 (cc) BY-NC-SA

Part 3 - Appendices

This page intentionally left blank

(cc) BY-NC-SA

Glossary

This is a comprehensive glossary of terms used in this manual

ADD. The size of the non-functional characteristics being added by an enhancement project

APM. Assessment Practice Manual

ASPA. The application SNAP Points after the enhancement project

ASPB. The application SNAP Points before the enhancement project

Assessment Categories. The framework on which the SNAP assessment is based

Assessment Criteria. Information used to determine the values used to develop the assessment

Assessment Questions. Questions that are related to specific attribute(s) which allows for the non-functional assessment of a given sub-category

Assessment Ratings. The answer to an assessment question using given criteria.

Assessment Scope. Defines the set of non-functional user requirements to be included in the assessment

Assessment Value ("SNAP Points"). Non-functional size measure (SP)

Category. A group of components, processes or activities that are used in order to meet the non-functional requirement.

CHG. The size of the changes made to existing sub-categories by an enhancement project

CHGA. The size of the non-functional requirements being changed by an enhancement project – as they are/will be after implementation

CHGB. The size of the non-functional requirements being changed by an enhancement project – as they are/were before the project commenced.

CPM. Counting Practices Manual

Decision Effective Date. Effective date of decisions to update the APM

DEL. The size of the non-functional characteristics deleted by an enhancement project.

DET. Data Element Type.
Unique, user recognizable, non-repeated attribute

DSP. Development Project SNAP points

EP. Elementary Process.
The smallest unit of activity that is meaningful to the user(s)

ESP. Enhanced Project SNAP points

FUR. Functional User Requirements.
A sub-set of the user requirements (UR); requirements that describe what the software shall do, in terms of tasks and services. (ISO 14143-1:2007)

IFPUG. International Function Point Users Group

Impact Study. A study that is initiated if there is any possibility that a common practice or several organizations or types of applications may change

NFR. Non-functional User Requirements.
A software requirement that describes not what the software will do but how the software will do it. [ISO/IEC 24765, Systems and Software Engineering Vocabulary.] Syn: design constraints, non-functional requirement. See also: functional user requirement

NFSSC. Non-functional Sizing Standards Committee

NFSSC Review. NFSSC reviews and discusses the rationale for each proposed update and its potential impact which will lead the committee to accept or reject the proposed update

Non-functional Assessment. Size of software in SNAP points

Non-functional Assessment process. Process described it the APM to arrive at a SNAP count

Partition. A set of software functions within an application boundary that share homogeneous assessment criteria and values.

RET. Record Element Type
User recognizable sub-group of data element types within a data function

ROI. Return on investment =([Gain from investment] – [Cost of investment]) divided by [Cost of investment])

SCU. SNAP Counting Unit
The component or activity, in which complexity and size is assessed.
The SCU can be a component, a process or an activity identified according to the nature of one or more sub-categories.
In some cases, the SCU is identical to the elementary process.

SNAP. Software Non-functional Assessment Process

SP. SNAP Points

Sub-Category. A component, a process or an activity executed within the project, to meet the non-functional requirement

User Requirements. Requirements describing what the user is asking for. (UR)

User View. A user view is the functional and the non-functional user requirements as perceived by the user

(cc) BY-NC-SA

IFPUG APM Link to IFPUG CPM

Introduction

This section describes how the Software Non-functional Assessment Process links to the Function Point Analysis process.

Caution: This is a preliminary view of the linkage between SNAP Points and Function Points. Further analysis is required to determine how the two size measures can be used together. Future releases will further address this issue.

Contents

This appendix provides the SNAP Process diagram and includes examples of potential SNAP uses.

Topic	Page
FPA and SNAP Link	B-2
Diagram of the Link between FPA and SNAP Processes	B-2
Counting Function Points and SNAP Points	B-2
Potential Uses of SNAP Points	B-13
Non-functional Size Measurement (SNAP Points)	B-14

 (cc) BY-NC-SA

FPA and SNAP Link

Non-functional size can be used in conjunction with functional size to provide an overall view of the project or application including both functional and non-functional sizing.

Assessing the effort impact on projects as a result of the SNAP size is out of scope of this document. Organizations should collect and analyze their own data to determine non-functional productivity impacts.

Potential uses of non-functional size together with functional size are provided by way of example.

Diagram of the Link between FPA and SNAP Processes

The following diagram illustrates the steps when sizing a project for both functional and non-functional requirements.

The purpose, scope, and logical application boundaries need to be consistent between the FPA and SNAP Processes.

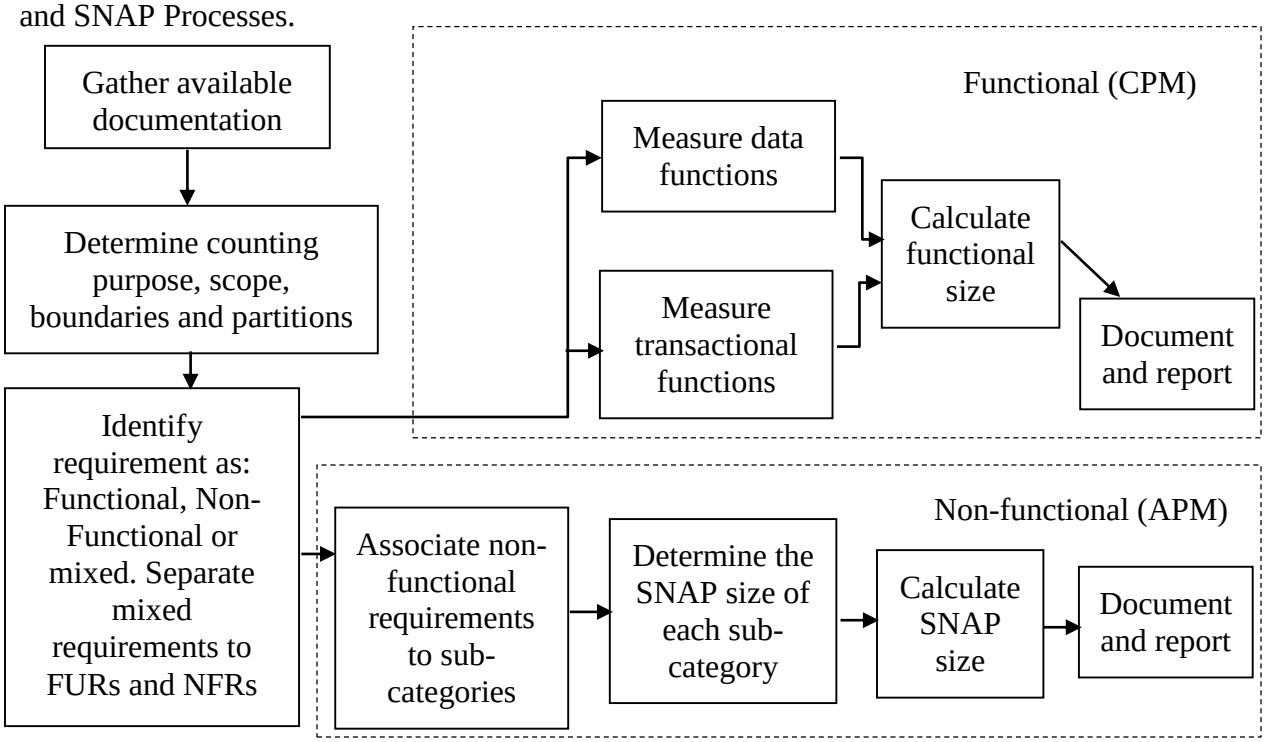

Gather available documentation
Determine counting purpose, scope, boundaries and partitions
Identify requirement as: Functional, Non-Functional or mixed. Separate mixed requirements to FURs and NFRs
Functional (CPM)
Measure data functions
Measure transactional functions
Calculate functional size
Document and report
Non-functional (APM)
Associate non-functional requirements to sub-categories
Determine the SNAP size of each sub-category
Calculate SNAP size
Document and report

Figure 3-1: The process of counting both FPA and SNAP

Counting Function Points and SNAP Points

A requirement may contain both functional and non-functional aspects. In such a case, the requirement will have a functional size, measured in Function Points, and a SNAP size, measured in SNAP Points.

Such a requirement should be broken down into its functional components and non-functional components, and the segregation should be agreed by both the user/customer and development teams. Use FP for FUR parts of the requirements and SNAP Points for the non-functional parts of the requirements.

(cc) BY-NC-SA

The following table is a guideline. To define NFR, ISO standard or a similar standard may be used.

Case #	Circumstance	Description	Guideline
1	Requirements are Functional Only	The users do not have any explicit or implicit non-functional requirements	Count function points only
2	Requirements are clearly marked as NFR	Parties agree on clear segregation between functional requirements and non-functional requirements; Requirements classified as NFR cannot be sized with function points	Count SNAP points only
3	Requirements involve both Functional and non-functional aspects	Functional requirements have additional NFR which can be clearly identified:	See table below
4	Requirements are functional only, transactions cross Partitions	Functional requirements may involve single or multiple flows. In case of multiple flows, and using the present CPM guidelines, each flow might not qualify as a separate elementary process.	Count function points to size the new/enhanced functionality for the main elementary process as per CPM, add SNAP size for the transactions / flows within the application's boundary, that cross the partitions
5	Requirements are functional, but they are provided without any software change	Functionality (or any business value) that is added or modified by changing reference data or other means that cannot be sized by function points, according to the present CPM guidelines or FP counting practices of the organization.	Count SNAP points using sub-category 1.5 – Delivering Functionality by Data configuration

Table 3-1 FPA and SNAP interrelations

 (cc) BY-NC-SA

Requirements Involve Functional and Non-functional Requirement

The following guidelines should be used to determine how FP and SNAP points should be counted

1.1 Data Entry Validation	
What to check	The elementary process
Rules	1. Count SNAP Points when: a. Adding a new data entry field with validation; b. Changing a data entry field and changing the validation; c. Changing the data entry validation logic. 2. When adding a data entry field, count EI per the CPM for the functional aspect and SNAP for the validation. Sub-category 1.1 sizes the validation per its complexity. 3. Data entry validation are counted in this subcategory only if it involves adding, changing or removing validation logic related to what data is being entered. It does not include size of any other conditional analysis which is part of the business rules. For example, adding a rule to validate that a numeric entry is not negative, in addition to validating that the entry is numeric and is decimal – is SNAP counted. Adding a valid value to an existing list without changing the validation logic does not generate SNAP Points in this sub-category (Adding a valid value should be counted under sub-category 3.2 Database Technology, as a database change)
Examples	1. A change in a date field: Former validation checked the format of DD/MMM/YY and now format should be DD/MMM/YYYY. 2. In a travel application for international flight booking function, passport type field is being added. A new code table is added to store passport type (regular, diplomatic, etc.) Based on passport type, validation rules exist to check the format of passport number being entered is correct or not. Count FP for adding the new field, SP (sub-category 1.1) for the data entry validation, and SP (sub-category 3.2) for adding the code table. 3. In a telecommunication self-service application, payment screen: Validate that the mobile number is numerical, and is structured as "0XX-XXXXXXX" (first digit must be zero, following 9 digits only) should be SP counted. 4. In a health insurance application, a claim has many fields. Validating that all fields are populated with the right format and that all mandatory fields are not empty are SNAP counted using this sub-category 5. A change is requested to the values in drop down list with 5 values, change 1 value and add 2 new values: If data entry validation logic does not change, use sub-category 3.2 and not this sub-category to count SNAP Points.

(cc) BY-NC-SA

1.2 Logical and Mathematical operations	
What to check	The elementary process
Rules	1. Count FP for any EO / EI / EQ per the CPM; in addition, count SNAP Points when processing logic includes extensive mathematical and / or extensive logical operations as defined in Part 1 Chapter 5 of this manual. The complexity of the algorithmic calculations and / or the logical processing required is the non-functional aspect of the requirement. 2. The above is true either for a development of a new request or for an enhancement.
Examples	1. A current algorithm required squaring up of (2/3rd exponent of the numbers) followed by their summation and then taking a square root of the result. As part of an enhancement, it is required to change the algorithm to squaring up of numbers followed by their difference and then taking a square root of the result. In this case, count both FP and SP. FP is impacted for processing logic change as there is change in calculation as per definition of CPM. SP needs be counted for assess the complexity change in the algorithmic processing being done.
1.3 Data Formatting	
What to check	The elementary process
Rules	1. When formatting is requested, break the requirements into its functional part (The "What") and its non-functional part (The formatting, such as encryption /decryption, byte padding, alignment / positioning, adding code data information). 2. New development may have functional aspect only (no requirements for any format) or both functional and non-functional aspects; Enhancement may have functional aspect only (Such as change in the data transferred) or non-functional aspect only or both.
Examples	1. To improve the look and feel of the report, the user requests that all the fields should add byte padding, to ensure a 20 character display field length for name field and 15 character display field length for the amount and the PID. The patient name should be displayed as First Name.Middle Name.Last name. In this case, the requirement should be broken into two parts – a) Functional: The patient name should be displayed as First Name.Middle Name.Last name. b) Non-functional:byte padding is added to ensure a 20 character display field length for name field and 15 character display field length for amount and PID. Count FP to size the transaction for part a): Display Report; Count SNAP

(cc) BY-NC-SA

	Points to size part b), using sub-category 1.3 2. A new HR system is being built to replace two outdated applications. To protect sensitive information, the user requests that the employee's social security number be masked on all display screens. When updating employee information, the user is provided with an option to unmask the social security number. In this case, the requirement should be broken into two parts – a) Functional: Display screens and Employee Update Screen should be counted under Function Points b) Non-functional: The masking and unmasking of the social security number is considered non-functional but should only be counted one time under SNAP. 3. The user has requested changes to existing display screens and one "Employee Update" Screen. In order to protect sensitive information the user requests that the employee's social security number be masked on all display screens. When updating employee information the user is provided with an option to unmask the social security number. In this case, there is only a non-functional requirement for the masking and unmasking of the social security number. Since there are no addition/deletion of data fields or change in the processing logic for the screens there is no change to the functional piece. a) Functional: None. b) Non-functional: The masking and unmasking of the social security number is considered non-functional and should only be counted using SNAP.
1.4 Internal Data Movements	
What to check	The elementary process
Rules	1. Count SNAP Points when partitions are defined, and the elementary process crosses partitions. (A new process as well as an enhanced process or deleted process). In addition, count FP. The FP counting measures the data movements in and out of the boundary, and the SNAP counting measures the internal data movements between partitions. 2. Count SNAP Points when a data flow between partitions is added, changed, or deleted to meet non-functional requirement. (See Example 2 below) Two scenarios can appear here: a) An elementary process (which delivers functionality on its own without invoking several other flows or processes) crosses multiple partitions. b) An elementary process that consists of multiple flows, which perform several tasks. These flows might perform different tasks of fetching, transforming, formatting and processing data, and creating outputs which

(cc) BY-NC-SA

	are finally consumed by the elementary process. These flows are not independent and user identifiable as per CPM guideline, and cannot be broken-up into several elementary processes.
Examples	1. An HR application consists of one application boundary with front end and back end as two partitions. A new data entry field is added and is used by three elementary processes: Add, change and enquire employee. All the employee related transactions receive input from the front-end application and processed with data from backend. Since there is a new field added, count FP for the functional change in the three elementary processes. In addition, count SNAP Points for the change in internal data movements in these 3 elementary processes. 2. An HR application consists of one application boundary with front end and back end as two partitions. A screen displays a value D, when D is calculated as A/ (B+C). A, B and C are retrieved from the back-end to the front-end, calculation is performed by the front-end. To improve performance, it is requested that the value D is calculated at the back-end; data flow to the front end is changed from (A, B and C) to (D). Count SNAP Points for the data flow change using sub-category 1.4.

1.5 Delivering Added Value to Users by Data Configuration

What to check	The elementary process
Rules	1. A software application may be designed in a way that functionality can be added or changed by adding or changing data in reference tables, and without any change to the processing logic as part of the application source code. According to the CPM, FP cannot be counted in this case. Count SNAP Points in such cases. 2. When functionality is added or enhanced using application source code change along with application configuration or addition of data to reference tables, and it generates FP, do not add SNAP size. 3. However, if the functionality added using application source code change and the functionality added using configuration are independent of each other, then count the functional change using FP, and the change added by configuration using SNAP.
Examples	1. CommWorld Company has an Application, which is designed to enable CommWorld products per geographical location. A new geography for product setup can be enabled by adding data to the "Function Profile" table. This table stores data on which product setup functions are available for which geography. The company has now expanded to Asia Pacific Area; for India location it needs to enable its product setup web pages for DSL packages; for Singapore it needs to enable its product setup web pages for DSL and Cable networks. Application development team creates one entry in "Function Profile" table for India location and two entries for Singapore location corresponding to

 (cc) BY-NC-SA

	each product. Since this change is enabled only by adding values to the table, FP cannot be counted. Use SNAP for sizing this change. 2. CommWorld Company has an Application which is designed to enable CommWorld products per geographical location. A new geography for product setup can be enabled by adding data to the "Function Profile" table. This table stores data on which product setup functions are available for which geography. The company has now expanded to Asia Pacific Area; for India location it needs to enable its product setup web pages for DSL packages; for Singapore it needs to enable its product setup web pages for DSL and Cable networks. For add or change product setup, the system is required to check internally that if same product package is being created in new geography as "already under sale in other geography", then corresponding row in "product package popularity" master table should also be updated. Application development team creates one entry in "Function Profile" table for India location and two entries for Singapore location corresponding to each product. Since, this change is enabled only by adding of values to the table, FP cannot be counted. For second part of the requirement, the new validation and update to product popularity master is a functional change to 'Add new product package transaction'. Hence it is FP countable. Since the reference table updates are accompanied with application source code change and they are not independent of each other, SNAP Points cannot be counted here, count only FP for the entire requirement.
2.1 User Interfaces	
What to check	Set of screens as defined by the elementary process
Rules	1. Creation of a new UI element in order to add, create or change functionality, which generates FP: Count FP and SP. (FP for the functionality, SP for configuring the UI element to meet non-functional requirements). Creation of a new UI element that does not add or change functionality (such as adding a static menu): Count SP only. 2. In case of modification of a UI element : a. If functionality is added or changed and also properties of the UI element are changed (See definition of properties in Part 1 Chapter 5), separate the requirements into its functional aspects (counted using FP) and its non-functional aspects (counted using SNAP). SNAP Points would assess the impacts to change in UI elements. b. If functionality is not changed, only properties of the UI element are changed (See definition in Part 1 Chapter 5) then count the change using SNAP.

(cc) BY-NC-SA

Examples	1. ABC Company merged with StarY Company. UI standards followed by the application development team have changed for font size, logo and background color. The application software needs to be enhanced to meet the latest version of User Interface standards. Use SNAP only to size the UI enhancement requirement. 2. A new field 'Customer Loyalty Points' is added to 'View Bill' transaction. Count FP for change to View Bill transaction for the newly added field. Count SNAP Points for the UI element impacts for the new field 'Customer Loyalty Points'.

2.2 Help Methods

What to check	The assessed application
Rules	1. Count SNAP Points for any types of "help" that is not Function Point counted. 2. Help techniques such as tool tips, dynamic help on mouse over (context help) are SNAP counted. 3. Static web pages are SNAP counted
Examples	1. A photo editing software application 'ZoomX' displays a number of photo editing options which are either available free of cost to user or under paid license. The requirement is to add tool tips to the existing photo editing tool icons, to display a message on the corresponding tool usage on mouse roll over. Count SNAP Points to size this requirement of adding tool tips. 2. Photo editing application had the pages 'About ZoomX', 'Contact us' which are static pages with a text on 'ZoomX' history and contact details respectively. Count SNAP Points to size these static pages.

2.3 Multiple Input Methods

What to check	The elementary process
Rules	1. Organization is using the Single Instance Approach: • Count Function Points per for the first input method only; • Count SNAP Points for each additional input method; • For new development requiring n input methods, count FP for one input method and SP for (n-1) input methods. 2. Organization is using the Multiple Instance Approach: • Count FP per for each input method; • Do not count SNAP Points.

 (cc) BY-NC-SA

Examples	1. The health care claim insurance software receives claim inputs via web screens or via batch inputs for paper claims. For organizations using single instance approach, shall count claim input via screen (first instance of same input type) using FP. All other similar inputs, like paper claim, would be counted using SNAP. Organizations using multiple instance approach, already accounted this complete functionality using FP. So SNAP cannot be applied in those cases.
2.4 Multiple Output Methods	
What to check	The elementary process
Rules	1. Organization is using the Single Instance Approach: • Count FP per for the first output method only; • Count SP for each additional output method; • For new development requiring n output methods, count FP for one output method and SP for (n-1) output methods; 2. Organization is using the Multiple Instance Approach: • Count FP per for each output method. 3. Do not count SNAP Points.
Examples	1. The health care claim insurance software sends processed claim output report via web screens or as paper output or as downloadable pdf. For organizations using single instance approach, shall count claim report output via screen (first instance of same output type) using FP. But all other similar outputs like paper report would be counted using SNAP. For organizations using multiple instance approach, shall have already accounted this complete functionality using FP. So SNAP cannot be applied in those cases.
3.1 Multiple Platforms	
What to check	The elementary process
Rules	1. For N platforms of any category (as defined in table 1-18) while N>1, count SNAP Points per table 1-18. 2. If N=1, do not count SNAP Points.
Examples	1. For a banking application written in JAVA, the 'my account view' transaction is required to be available on 3 different browsers - Internet Explorer, Chrome and Fire Fox in same format. The first platform deployment is FP counted; count SNAP Points for required transaction compatibility on the two other platforms, using this category. For a banking application written in JAVA, the 'Export Customer Data' transaction is executed every night cycle. This transaction is a functional requirement in itself and is counted using FP. SNAP cannot be counted for it. If the same transaction had to be re-written in VB to support any legacy system for

(cc) BY-NC-SA

	the same application boundary, then SNAP shall be used to size the additional platform support.

3.2 Database Technology

What to check	The elementary process
Rules	1. A database change, as defined in Part 1 Chapter 5 and according to the CPM, is not counted as an ILF or an EIF, therefore it is a non-functional change. It may be used either in addition to functional changes or to meet a pure non-functional requirements. In both cases count SNAP Points only.
Examples	1. Adding error code and description to a code table, while adding code that will generate these error. Count SNAP for the table changes. 2. User has requested a performance improvement for "search business partner" transaction to support B2E (Business-to-Enterprise) transactions. Development team decided to create a secondary index file on 'Business Partner Name' column on "Partner" Table, to improve search speed. Count SNAP Points using this category for the new index file being created.

3.3 Batch Processes

What to check	The user identified batch job
Rules	1. When a user identifiable batch job is not qualified as any of the transaction functions per FPA guidelines, use SNAP to size the batch jobs. 2. When an existing batch job is divided into multiple jobs and the reason is to meet non-functional requirements, count SNAP Points to size the batch jobs.
Examples	1. The marketing data warehouse has a user requirement for automatic archival of marketing campaigns older than 5 years. An end-of-month job archives data based on this criteria. The job reads data from within the database and after applying the necessary logic, moves the data to archive tables within the same database without any data crossing the boundary as input or output. Count SNAP Points for this requirement. 2. In a travel application, user is given a provision for cancellation of a ticket. Post cancellation, the system automatically un-applies the booking receipt and initiates a new request for credit back to the user, by making an entry in Credit table and adding a task for customer service agent to issue the credit note. As per CPM, any transaction which does not cross the application boundary is not FP counted. Counting teams may have assumed auto triggers as the DET crossing the boundary and accounted for this transaction in function point. In such case, where auto trigger is assumed as part of local FP guidelines, do not count SNAP Points. Else, count the above requirement using SNAP.

(cc) BY-NC-SA

4.1 Component Based Software	
What to check	The elementary process
Rules	1. This subcategory adds non-functional size to the counted functionality, when the software comprises of re-usable components. 2. Count FP per the CPM per each EP. 3. In addition, count SNAP Points when the EP consumes components
Examples	1. A Travel Application is required to build components which can be re-used across different travel products (such as hotel and flight) for booking a product type. All the components are build in-house. Reusable components include the following a) GetPassengerDetails b) ProcessPayments c) SendUserNotificiation Each of the product type would have its own elementary process for booking (book a flight, book a hotel) which would use the above re-usable components along with other processing logic which is unique to that process. Count SNAP Points to size the respective elementary process for each product type. Count SNAP Points to size the requirement for re-usable components to be created.
4.2 Multiple Input / Output Interfaces	
What to check	The elementary process
Rules	1. Count FP for creating the functionality. If a new development or a new requirement includes multiple inputs and / or outputs, count FP for the functional aspect and SP for the additional interfaces 2. When a functional change is required, count FP for the change (as if one interface exists) and add SP for the additional number of interfaces (according to table 1-23) 3. When there is no functional change and the only change involves adding or removing interfaces, count SNAP Points using this sub-category
Examples	1. An online retail application is required to support increased volume of user transactions for its mega discount offers on Christmas. Application development team decided to add new servers to support the increased volume. Use SNAP for sizing the setup required for all processes enabled via use of additional servers.

Table 3-2 FPA and SNAP interrelations rules

(cc) BY-NC-SA

Notes:

1. SNAP introduces an additional non-functional size to functionality that has been sized before, using function points only. Therefore, the overall size of the application includes now a separation between the functional aspects of the non-functional aspects.
2. A requirement must be separated into its functional aspects and its non-functional aspects. The functional aspects are sized using function points and the non-functional aspects are sized using SNAP.

Potential Uses of SNAP Points

Project Estimation

Project Estimation

- Count FPs for project – utilize productivity rates based on FP size and project type to estimate functional project effort etc., from data contained in repositories.
- Complete SNAP assessment – utilize historical data to determine impact of SNAP score on project effort. Adjust project effort up or down depending on SNAP score to provide a functional/technical effort estimate.
  The relationship between SNAP and effort may be specific to an organization.
  Once significant SNAP assessment data is collected, a "rule of thumb" relationship may be established by industry, platform, etc, from data contained in repositories.
- Complete risk/attribute Assessment – Assesses organizational factors that impact productivity. Utilize historical data on risk/attribute impacts on effort and adjust project effort up or down appropriately to provide a functional/technical/risk effort estimate.

Note: Can use all three effort estimates individually or in any combination to produce an estimate range. The productivity ratio for FP should reflect only the effort required to implement the functional requirements and the SNAP productivity ratio should reflect only the effort required to implement the non-functional requirements.

ROI

Calculate Return on Investment (ROI) Estimate of Application Replacement

- Count FPs for Application to determine functional size of the application
- Complete SNAP assessment to determine the non-functional size of the application

Compare cost of the replacement project and the future maintenance costs with the maintenance cost of the existing application to determine the ROI.

 (cc) BY-NC-SA

Emphasis Area(s)

Utilize SNAP to indicate the non-functional areas of emphasis in the project/application. Examine the detail responses to the SNAP assessment compared to ISO/IEC 25010 characteristics (or another standard for classifying non-functional requirements) to ensure the appropriate focus

Example: a project with more SNAP Points allocated to accuracy versus attractiveness indicates that there is more emphasis on accuracy.

Options Comparison

Comparison of non-functional alternatives by having an overview of total costs (functional and non-functional)

Examples:

- Using COTS versus development
- Compare technologies that deliver functionality in order to select the appropriate technology (e.g., database technologies, interface technologies)

Maintenance Cost

Assist in overall assessment of maintenance costs/resources

- Use FP to size the functionality of the application
- Use SP to size the non-functional characteristics of the application

Use historical data to estimate the maintenance costs by functional and non-functional size for each application to plan future resources and aid in maintenance strategies and annual budgeting.

Non-functional Size Measurement (SNAP Points)

The IFPUG CPM has been transformed into an ISO standard for functional size measurement with the exclusion of the General System Characteristics (GSCs), which assess the general functionality of the application. (IFPUG FPA V4.1 unadjusted was received as ISO standard from 2003 - ISO/IEC 20926:2003 - later in 2009 for v4.3). The most important consideration in relation to these issues is how the non-functional requirements affect the size. Since they are not functional requirements, they do not contribute to the functional size. However, they are still part of the overall requirements (functional and non-functional) for the software, and therefore contribute to the overall size of requirements.

SNAP have been developed to be used independently of the GSCs to assess the non-functional requirements. Practitioners of the SNAP Point assessment should NOT use both SNAP and GSCs simultaneously. Using both may inflate the resulting non-functional size of the non-functional requirements. SNAP can be used in conjunction with function points (given identical boundaries) to provide an overall view of a project or application including both functional and non-functional sizing.

(cc) BY-NC-SA

Index

application
- Part 1, 1 -, 8
- Part 1, 2 -, 3
- Part 1, 2 -, 2
- Part 1, 2 -, 8
- Part 1, 3 -, 3
- Part 1, 4 -, 2, 3, 4
- Part 1, 5 -, 7, 10
- Part 1, 6 -, 50, 51
- Part 2, 1 -, 4, 6, 18
- Preface -, v, x, xiii, xiv

Architecture
- Part 1, 2 -, 6
- Part 1, 3 -, 3
- Part 1, 5 -, 39

ASPA
- The application SP after the enhancement project
- Part 1 6 -, 51, 52
- Part 1, 4 -, 3
- Part 1, 6 -, 51

assessment type
- Part 1, 4 -, 2

Assessment Type Definition
- Part 1, 2 -, 3

Batch Jobs
- part 2, 1 -, 16

Batch processes
- *Examples Part 2* -, 16
- Part 1, 5 -, 7, 37

Boundary
- Boundary definition -Part 1, 4 -, 5
- Boundary Rules -Part 1, 4 -, 7
- Part 1, 2 -, 3, 4
- Part 1, 4 -, 1, 2, 5, 7
- Part 1, 5 -, 18
- Part 1, 6 -, 54

Boundary and Partitions Hints
- Part 1, 4 -, 7

Calculate Non-Functional Size
- Part 1, 2 -, 8

Category
- Architecture - Part 1, 5 -, 39
- Data Operations - Part 1, 5 -, 10
- Interface Design - Part 1, 5 -, 22
- Technical Environment - Part 1, 5 -, 31

Category Definition
- Part 1, 2 =, 5

Certification, xvi

Code Data
- Part 1, 5 -, 4, 11, 20

Complexity
- definition - Part 1, 5 -, 3
- Example - Part 1, 5 -, 3
- grid - Part 1, 5 -, 3
- Parameters - Part 1, 5 -, 3

Component Based Software
- Part 1, 5 -, 39
- *Part 2, 1* -, 17

Consistency with FPA
- Part 1, 4 -, 2

Data Element Type (DET)
- Part 1, 5 -, 7

Data entry validation
- Part 2, 1 -, 2

Data Entry Validation
- Part 1, 5 -, 10

Data Formatting
- Part 1, 5 -, 16
- Part 2, 1 -, 3

Data Operations
- Part 1, 2 -, 6
- Part 1, 5 -, 10

Database Technology
- Part 1, 5 -, 35
- Part 2, 1 -, 4, 15

Database View
- Part 1, 5 -, 35
- Part 1, 5 -, 8

Delivering added value to users by Data Configuration
- Part 1, 5 -, 20

Delivering Added Value to Users by Data Configuration, 7

Development Project
- Part 1, 2 -, 3
- Part 1, 4 -, 3

Part 1, 4 -, 3
Part 1, 1 -, 9
Part 2, 2 -, 2
Preface -, vi

Documentation, xiv
Guidelines for Software Measurement, xiv
IFPUG, An Introduction, xiv

DSP - The development project SP
Part 1, 4 -, 3
Part 1, 6 -, 49

Elementary Process (EP)
Part 1, 5 -, 7

Enhancement Formula Rules
Part 1, 6 -, 50

Enhancement Project
Part 1, 6 -, 54
Part 1, 2 -, 3
Part 1, 4 -, 3
Part 1, 6 -, 50, 51, 53
Part 2, 2 -, 2

ESP - The enhancement project SP
Part 1, 4 -, 3
Part 1, 6 -, 50

Estimated and Final Non-Functional Assessment
Part 1, 3 -, 4

Examples
Adding Input and Output Methods Part 2 -, 11
Batch Jobs Part 2-, 16
Change free text to valid values Part 2 -, 2
Compliance with Standard Part 2 -, 8
Component Based Software Part 2 -, 17
Creating new Products and Offers Part 2 -, 6
Data Security Part 2-, 3
Help Part 2-, 10
Internal Data backup and data transformation Part 2 -, 4
Multiple Interfaces Part 2 -, 18
Multiple Platforms Part 2 -, 13
Performance Improvement Part 2 -, 14
SNAP Calculation Case study
Part 1, 6 -, 53

Formula
application size after enhancement project - Part 1, 6 -, 51
SP for development project - Part 1, 6 -, 49
SP for enhancement project - Part 1, 6 -, 50

Formula Approach
Part 1, 6 -, 49

FTR (File Type Referenced)
Part 1, 5 -, 14
Part 1, 5 -), 8

FTR Complexity
Part 1, **5 -**, 36
Part 2, 1 **-**, 5

Functional User Requirements
Part 1, 1-, 3

Help Methods
Part 1, 5, 25
Part 2, 1 **-**, 10

IFPUG documentation, xiv

Interface Design
Part 1, 2 -, 6
Part 1, 5 -, 22

Internal Data Movements
Part 1, 5 -, 18
Part 2, 1 -, 4

ISO/IEC 14143-1
Part 1, 1-, 3

ISO/IEC 25010
App B -, 16
Part 1, 1 -, 6
Part 1, 1 -, 4, 6
Part 1, 1-, 3
Part 1, 1 -, 2

ISO/IEC 9126-1
Part 1 1 -, 6
Part 1, 2 -, 6
Part 1, 1 -, 6

Logical and Mathematical operations
Part 1, 5 -, 13

Logical file
Part 1, 5 -, 8, 14, 15, 20, 36
Part 2, 1 -, 6

Manual
Change Process, xii
final decisions, xiii
Frequency of Changes, xii
how decisions are communicated, xiii

Mission Critical
Part 1, 5 -, 45
Part 1, 5 -, 45

Mission Critical / Real Time Systems
Part 1, 5 -, 45

Multiple Input / Output Interfaces
Part 2, 1 -, 18
Multiple Input Methods
Part 1, 5 -, 28
Part 2, 1 -, 11
Multiple Instance approach
Part 1 5 -, 29
Part 1, 5 -, 9, 30
Multiple Output Methods
Part 1, 5 -, 29
Part 2, 1 -, 12
Multiple platforms
Part 1, 5 -, 31
Multiple Platforms
Part 2, 1 -, 13
Non-Functional User Requirements
Part 1, 1-, 4
Part 2 Examples
SP Calculation Example
Part 1, 5 -, 46
Partition
Part 1, 2 -, 3, 4
Part 1, 4 -, 6
Part 1, 5 -, 10
Part 1. 2 -, 3
Part 2 (Examples), 4
Real time
Part 1, 5 -, 2, 32, 45
Record Element Type (RET)
Part 1, 5 -, 36
Part 1, 5 -, 8
Part 2 (Examples) -, 14
Scope hints
Part 1, 4 -, 6
Single Instance approach
Part 1
5 -, 28
Part 1, 5 -, 9, 30
SNAP Benefits
Part 1, 1 -, 11
SNAP Counting Unit (SCU) Definition
Part 1, 2 -, 7
SNAP Framework
Part 1, 1 -, 8
SNAP Objectives
Part 1, 1 -, 10
SNAP Points (SP)
Part 1, 2 -, 8
SNAP Procedure by Section
Part 1, 2 -, 2
Software Component
Part 1, 5 -, 39
software project
Part 1, 5 -, 3
Part 1, 1 -, 8
Part 2 -, 13
SP
Part 1, 2 -, 1, 8
Part 1, **3 -**, 2
Part 1, 4 -, 3, 1
Part 1, 5 -, 3, 8, 10, 15, 16, 18, 20, 23, 28, 29, 32, 36, 37, 40, 41, 45
Part 1, 6 -, 50, 54
Part 2, 1-, 13
Part 2, 1 -, 2, 3, 5, 7, 9, 10, 12, 15, 16, 17, 19
Part 2, 6 -, 49
SP Calculation Example
Part 1, 5 -, 46
SPApp A, -, 1
SPPart 1, 5 -, 25
Sub-category
Batch processes - Part 1, 5 -, 7, 37
Component based software - Part 1, 5 -, 39
Data Entry Validation - Part 1, 5 -, 10
Data Formatting - Part 1, 5 -, 16
Database Technology - Part 1, 5 -, 35
Delivering added value to users by Data Configuration - Part 1, 5 -, 20
Help Methods - Part 1, 5 -, 25
Internal Data Movements - Part 1, 5 -, 18
Logical and Mathematical operations - Part 1, 5 -, 13
Mission Critical / Real Time System Part 1, 5 -, 45
Multiple Input Methods - Part 1, 5 -, 28
Multiple Output Methods - Part 1, 5 -, 29
Multiple platforms - Part 1, 5 -, 31
User Interfaces
Part 1, 5 -, 22
Sub-category Complexity
Part 1, 5 -, 3
Sub-category Definition
Part 1, 2 -, 6
Technical Environment

(cc) BY-NC-SA

Part 1, 2 -, 6
Part 1, 5 -, 31
Timing of Non-Functional Assessments
Part 1, 3 -, 2
Training Requirements, xvi
Type of Assessments
Part 1, 4 -, 3
UI Element
App. B -, 10
Part 1, 5 -, 22, 23
Part 2, 1 -, 8
Useful Project/Application Documentation
Part 1, 3 -, 3
User Interfaces
Part 1, 2 -, 6
Part 1, 5 -, 22, 23
Part 2, 1 -, 9
Part 2, Examples -, 8
User Requirements
Part , 1 -, 3

Part 3 Appendix D

APM Changes after Release 1.0

Release 2.0, January 2013

The second public release of the SNAP APM (January 2013) refines the concepts, processes, and rules for assessing non-functional software requirements.

More definitions have been added to clarify the terms used in this manual.

Some sub-categories were re-defined based on users' experience and comments.

Guidelines have been added to refine the linkage between FP and SNAP, and to detail how to size requirements that involve both functional and non-functional aspects. Users should verify that there is no duplicated sizing; SNAP is complementary to FPA and not a replacement.

Release 2.1, April 2013

The third public release contains more clarifications, based on questions and comments of users.

In addition, the sizing of Coda Data was inserted into the relevant sub-categories.

More clarifications were added to the sub-categories:

- Sub-category 1.2 Logical and Mathematical Operations, to explain the term "extensive mathematical operations"
- Sub-category 1.5, the SCU definition now includes the case of several logical files (as shown in Example 4). Tables 2-6 and 2-7 were modified to better explain how SP calculation is done.
- Sub categories 2.3 and 2.4 Multiple Inputs and Output methods, sub-category 4.2 Multiple Input / Output interfaces: Clarifications were added to the complexity parameters to clarify that SP size derived from the number of additional methods or interfaces.
- Clarifications were added to Table 2-16 and 2-17 to the counting formula.

Release 2.2, April 2014

This release contains additional clarifications and error corrections.

- Adding SNAP certification paragraph to the Introduction chapter.
- Removing "the batch function" and "single application" from the examples of partition.
- Adding brackets (ling to FPA) to Boundary and Partition Hints.
- Clarifications to the definition of "Logical file."
- Definition of "database view" was added.
- In sub-category 1.2, name was replaced from "FTR complexity" to "FTR density."
- A note was added to 1.2.
- Clarification to SOA was added to notes in sub-category1.4.
- Definition of Logical Files moved to "Terms."
- "Labels" added as a UI element to sub-category 2.1, and an example was added.

(cc) BY-NC-SA

- Note added to 2.2 stating that calibration of this sub-category remains open for future research.
- Calculation example in sub-category 3.1 was corrected.
- Example in sub-category 3.2 was corrected.
- The note dealing with code data was removed from 3.2.
- In figure 1-5, boundary was added. Header changed from hotel reservation to holiday reservation.
- A note was added to sub-category 4.2.
- Table 1-25 was fixed.
- Example 1, 6 and 9 were expanded and SP calculations were changed.
- Example 4, Tables 2-6 and 2-7 were changed (more accurate terms).
- Example 8 was fixed, to comply with the second note on page 5-30.
- Clarifications added to figure 2-1.
- In example 12, SP calculations were corrected.
- In table 3-2, for 2.2 Help Methods, the "what to check" field was corrected.
- Terms added to the glossary (ASPA, DET, DSP, and NFR). "Technical size framework" and "Data element type" removed from glossary.

Release 2.3, May 2015

This release contains additional clarifications and error corrections.

- The release notes section (This section) was moved from part 1 to Appendix D (This appendix).
- Note on section 1 "Partition"– added reference to sub-category 1.4.
- Sub-category 1.1: Clarification added regarding unique DETs (Word "unique" added instead of "maximum number of DETs..").
- Examples were added.
- Sub category 1.4:
    - The definition of the sub-category and the SCU were improved (the changes clarify the definition but do not change their meaning).
    - Notes were added with more examples and better explanation of how SNAP is used.
    - An example was added.
- Sub-categories 2.3 Multiple Input Methods and 2.4 Multiple output Methods:
    - Note was added ("When counting a new development project, the number of input methods should include the additional ones only…").
- Sub-category 3.1:
    - Notes were added to clarify SP counting in a new development / new requirements and enhancements.

  - Notes and examples were added to demonstrate SP calculation when multiple platforms from multiple categories are involved.

- Sub-category 3.2:
  - The term sub-item was added, to better clarify what is considered as a database change.
  - Clarification added to the list of sub-items
  - A note was added to clarify how to count SP in a new development / new requirement as well as enhancement.
- Formulas to determine the Non-functional size for enhancement project:
  - This chapter was edited to improve the clarity.
  - Formula shown was per category, now it is per sub-category (following counting practice).
  - The definition of "Change" was improved. The term CHGA was changed to CHG, to separate CHG in project count from CHGA for application count after an enhancement project.
  - Note was added ("The size of changes made…").
  - Two examples were added.
- Sub-category 4.2: A simple example was replaced by Examples 1 to 4 and consequently, figure 1-6 was replaced by figures 1-6.1, 1-6.2, 1-6.3, 1-6.4, 1-6.5, 1-6.6 and 1-6.7.
- Table 1-24: SP calculations was corrected.
- All examples are now with a common format of: Requirements; Analysis; Counting.
- Example 4: The explanation of the 2 processes that are counted (out of 4 possible processes) was improved.
- Example 9: Table 2-13: Number of DETs corrected.
- Glossary: Format was changed from one column to two columns (same as CPM format).
- Appendix B, FPA and SNAP link:
  - Figure 3-1 was modified. Previously the functional and non-functional measurements processes were independent; now the first 3 activities are common for the functional and non-functional measurements processes.
  - Table 3-2 was enhanced: For each sub category, a set of rules were added as well as examples, clarifying how to count FP and SP without duplication.
- Missing terms were added to the Glossary: ADD, ASPB, CHG, CHGA, CHGB, DEL.

 (cc) BY-NC-SA